Project Report

On

“ROOM ON RENT”

Submitted By

“YEOTIKAR PRASHANT SHIVRAJ”

“PATIL ASHUTOSH ANIL”

MASTER OF SCIENCE IN COMPUTER SCIENCE



School of Computational Science

SWAMI RAMANAND TEERTH MARATHWADA UNIVERSITY

NANDED (M. S.) 431606

Year 2021-2022

Project Report

On

ROOM ON RENT

Submitted By

YEOTIKAR PRASHANT SHIVRAJ

[VZ90378]

PATIL ASHUTOSH ANIL

[VZ90371]

Guided By

Prof. M. R. MAHAMUNE

In partial fulfilment for the award of

MASTER OF SCIENCE IN COMPUTER SCIENCE

School of Computational Science

SWAMI RAMANAND TEERTH MARATHWADA UNIVERSITY

NANDED (M. S.) 431606

Year 2021-22

**CERTIFICATE**

This is to certify that, the project **“ROOM ON RENT”** submitted by

**“YEOTIKAR PRASHANT SHIVRAJ”**

Is a Bonafide work completed under my supervision and guidance in partial fulfilment for

award of Master of Science in Computer Science Degree of Swami Ramanand Teerth Marathwada University, Nanded.

Place :- Nanded

Date :- 31 May 2022

Dr. M. R. Mahamune Dr. S.D. Khamitkar

Guide Director

**CERTIFICATE**

This is to certify that, the project **“ROOM ON RENT”** submitted by

**“PATIL ASHUTOSH ANIL”**

Is a bonafide work completed under my supervision and guidance in partial fulfilment for

award of Master of Science in Computer Science Degree of Swami Ramanand Teerth Marathwada University, Nanded.

Place :- Nanded

Date :- 31 May 2022

Dr. M. R. Mahamune Dr. S.D. Khamitkar

Guide Director

**INDEX**

1. [INTRODUCTION](#_TOC_250000)
   1. Introduction

1.2 Necessity

1.3 Existing System and Need for System

1.4 Scope of Work

1.5 Objectives

2**. ANALYSIS**

# **PROPOSED SYSTEM**

* 1. Proposed System
  2. Objectives of system

3.3 User Requirements

# **SYSTEM DEVELOPMENT**

* 1. DFD DATA FLOW DIGRAM
  2. Entity Relationship Diagram (ERD)
  3. Front End Design, Menu Tree, Menu Screens, Input Screens
  4. Coding

# **PERFORMANCE ANALYSIS**

* 1. Testing
  2. Implementing Testing

# **CONCLUSION**

* 1. Conclusion
  2. Future Scope
  3. Applications/Utility
  4. User Manual
  5. Operations Manual / Menu Explanation
  6. Forms and Report Specifications
  7. Drawbacks and Limitations

6.8 Proposed Enhancements

**REFERENCES**

**ANNEXURES**

**ACKNOWLEDGEMENT**

**1. INTRODUCTION**

**1.1 Introduction**

An Online Room Rental System **“ROOM ON RENT”** will provide the Information about Rooms/Flats/Houses which is available for Rent. It will also give information about hostels, like boy’s hostel and girl’s hostel. Room On Rent application will make easy to find the location of Rooms/Flats/Houses, select no. of rooms and other facts by the renter.  It will make easy to upload the location, contact No., expected rent, No. of rooms, Facilities and other information by landlord/room-master. This application will be able to show the rooms/flats/houses with in a particular area selected by user. It is also difficult to find the renter on time, for the landlord and property managers. It is also difficult to find the renter on time, for the landlord and property managers. Recently This Kinds of system working in New York, Boston, or United States.

**1.2 Necessity**

Room on rent system has become important factor in modern society hence the need to have a rental room finder application. It is difficult to find house for the renter in a specific area.

Many students move from their houses to another cities for their studies. Migrant workers move from one state to another for jobs. Different kinds of jobs and fields are their and along with that people also come from different places. They start finding the place to stay and the process of finding a place to stay is very hard and time consuming. Generally people from another place go to some another place and they don’t know the area is safe or not, they may face different problems from finding a place to living at that place.

Due to these problems a need of online room finder has emerged. The necessity of room on rent is very obvious in current time. This research work was undertaken to uncover the various problems with conventional room searching. The conventional way of finding a rental room is difficult and time consuming. Generally people don’t know the locality around room is safe or not.

**1.3 Existing System and Need for System**

There are various applications which provide services for rental rooms, hostels, houses. Websites like Magic bricks, 99acres.com, room, nest away give services regarding rental rooms and houses. These systems are highly complicated and confusing to understand. Our software can do the same work or better work than these software’s. With its few segments it is easy to use, easy to understand, it’s very useful. The people want to search room, the students who want to search room they can use this site.

**1.4 Scope of Work**

Housing plays a huge role in revitalizing economic growth in any country, with shelter being among the key indicators of development. Most families, choose to rent houses based on their income and family situations. Unfortunately, there may not be enough good quality rental housing for these families. The demand for rental houses is extremely high and more rental houses need to be put in place. Developing rental houses has numerous advantages, particularly for landlords, who can increase their profits through the rent paid by tenants. The increased number of tenants and landlords complicates management, especially for landlords who lose large sums of money to tenants who do not pay their rent. The above statement gives a clear declaration as to why rental house management system needs to develop.

**1.5 Objectives**

The very first objective here is to reduce the efforts of paying guests and student to find rental rooms at their desired locations.

Finding perfect match of students, customers and the land lords on the basis of various filters like -

Rental amounts.

Peaceful areas.

Veg - Non veg issues and availability of resources.

Many people are not aware about rooms nearby their work place so they exploit petrol to travel and help raising economic and environmental issues.

So our software is going to solve many frustrating tasks at one place peacefully.

And we hope people will feel our software useful and address their issues properly. For that matter we have also added the feature of review and feedback for helping customers in making our software more reliable and convenient for them.

**2. ANALYSIS**

Room on rent is a place where students and other people can search and see the rental rooms for their accommodation. Searching a room is a tough task and this software will help to ease the process of finding a rental room. The product is a web application used to search rental rooms, hostels for girls and boys, houses. Objective of the system is to provide a user friendly environment where they can find there accommodation place easily. Our primary focus is to develop a effortless system that provides the way to search and see the rental rooms for students.

The first phase of this project is requirement gathering. This project needs a page where rooms will be displayed, there will be information about the room. The project needs 10-12 days of time to develop. It will be tested after its development finishes.

The resources required are computer enabled with os, it should contain pycharm, browser, internet, SQLite.

The time required to complete the development is 10-12 days. The time schedule is given in the Gantt chart of the synopsis.

There could be some problems during project development. Time is the only problem for this project. Finishing this project on time is a tough task.

To create this project vs code, SQLite, browser, internet and python is required.

**3. PROPOSED SYSTEM**

**3.1 Proposed System**

In our proposed system we have the provision for adding the rooms of the people by themselves. They can add the room and showcase their room on site. Another advantage of the system is that it is very easy to edit the details of the room and delete a room when it found unnecessary. The records of the room are added in the database and so user can also view the data whenever they want.

Our proposed system has several advantages

* User friendly interface
* Fast access to database
* Less error
* More Storage Capacity
* Search facility
* House images
* User-friendly environment

All the manual difficulties in searching room or renting room have been rectified by implementing computerization.

**3.2 Objective of the system**

The very first objective here is to reduce the efforts of paying guests and student to find rental rooms at their desired locations.

Finding perfect match of students, customers and the land lords on the basis of various filters like -

Rental amounts.

Peaceful areas.

Veg - Non veg issues and availability of resources.

Many people are not aware about rooms nearby there work place so they exploit petrol to travel and help raising economic and environmental issues.

So our software is going to solve many frustrating tasks at one place peacefully.

And we hope people will feel our software useful and address their issues properly. For that matter we have also added the feature of review and feedback for helping customers in making our software more reliable and convenient for them.

**3.3 System Requirements**

SOFTWARE SPECIFICATION

 WINDOWS 10

 PYCHARM

 PYTHON

 SQLITE3

HARDWARE SPECIFICATION

 INTEL® CORETM i5, 3.20GHz

 4 GB RAM

 256 GB STORAGE SPACE

 LAN CARD

**4. SYSTEM DEVELOPMENT.**

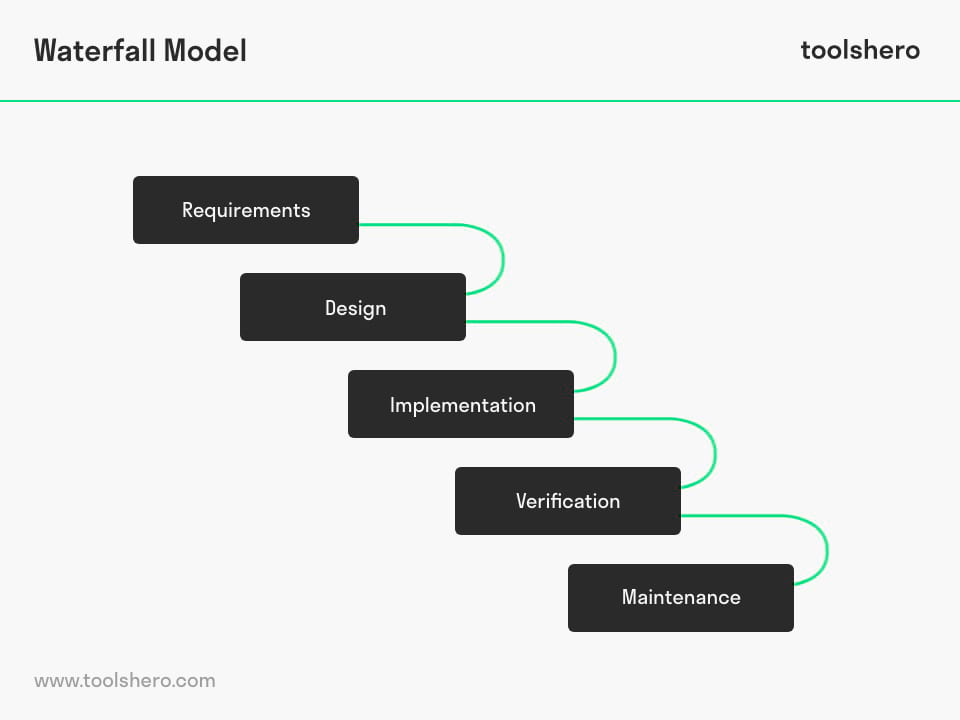
**4.1 Which SDLC Model is Used?**

WATERFALL MODEL

When applied in software development, the model is an iterative and flexible approach where progress, like the water of a waterfall, is largely in a single downward flow through the stages. The phases are: conception, initiation, analysis, design, construction, testing, implementation and maintenance.

The waterfall method is the first process model ([SDLC](https://www.toolshero.com/information-technology/systems-development-life-cycle/)) that was introduced. Originally, the waterfall method started in the construction industry.

Highly structured and physical environments meant that design changes during construction resulted in sky-high costs. Subsequently, when the demand for software increased, there was no recognized alternative to managing these projects.



### System and software requirements: The first thing is what we want to build? The answer is we want to create a web application that will help the students to find a rental room. We hope that this web app will be enough to solve many problems of searching rental rooms. If this application helps little bit then we want to add many new features in this web application in future. The system on which this software will run must have at least 4 GB ram, 200 GB hard disk space, internet connection, operation system.

### Analysis: The information gathered in the first phase is used here to generate product models and logic of room on rent application. This is very important for the management of production. A feasibility test is also done at the end of this phase. This applies to the financial and technical resources.

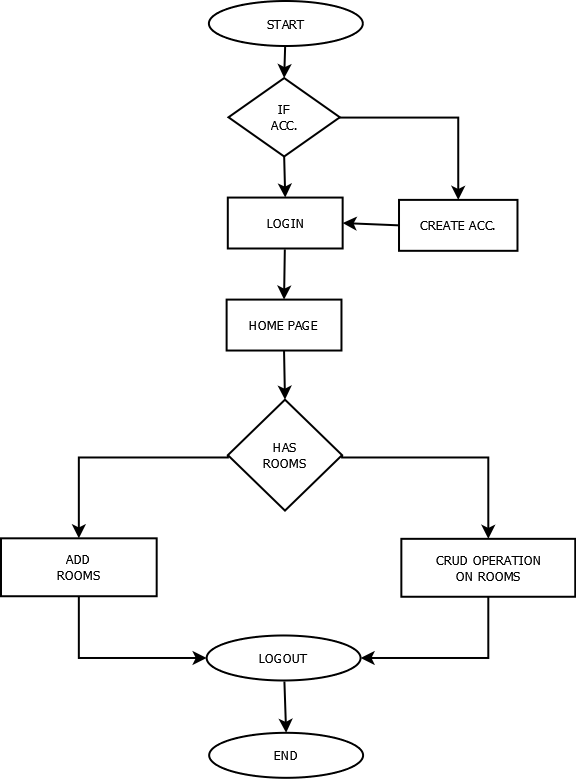
### Design: The third phase is the design phase. During this phase, the specifications from the first phase are studied and a system design of room on rent is prepared. The system design helps with establishing the system- and hardware requirements of room on rent.

1. Thereafter it helps with the definition of the general system architecture. This phase must be completed before going ahead with the coding phase. The code that will be developed in the next phase is in fact being prepared now. The result of this phase is a High-Level Design Document (HLD), or a Low-Level Design Document (LLD).
2. Coding: In the fourth phase, the code of room on rent is developed using the product models, logic and requirements from the previous phases.

### Testing: After coding is completed the system design is tested. This is to ensure that there are no errors in the room on rent system when the client starts using it. A variety of tests are applied depending on the project. Think of quality assurance, system tests, beta tests or unit tests.

1. If the system passes all the tests, the waterfall will continue to descend. The result of this phase is usually a test report, but in some cases also consists of a User Acceptance Test (UAT).

**4.2 System Flowchart**

****

4.3 **DFD**

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).  
On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.  
A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

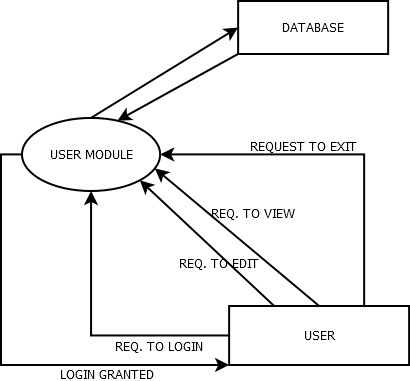
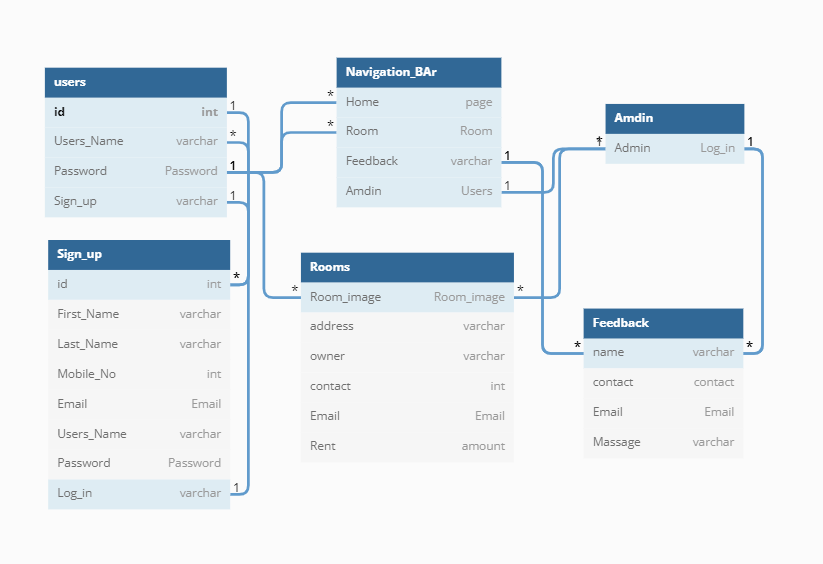


DIAGRAM DFD DIAGRAM OF ROOM ON RENT

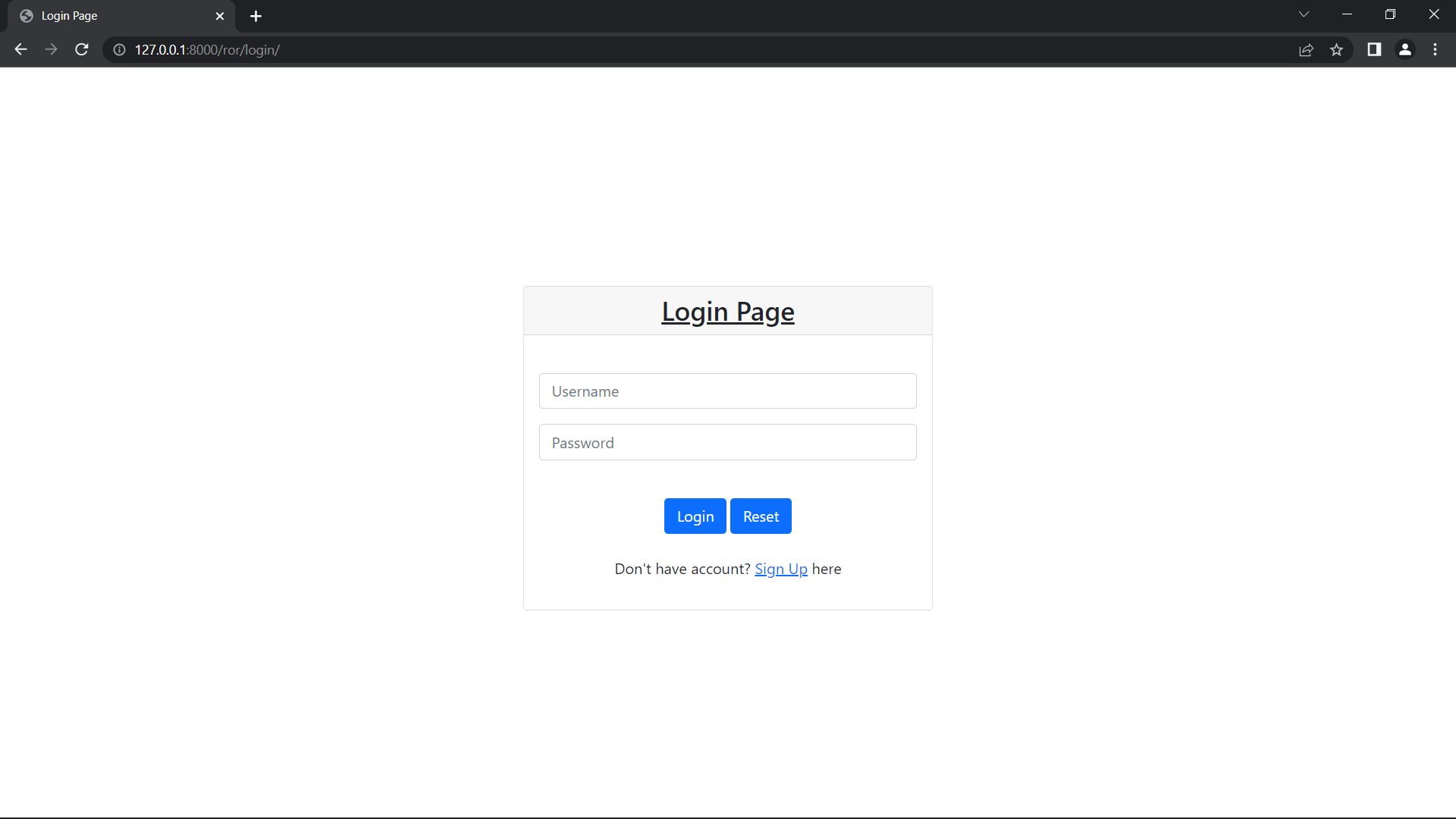
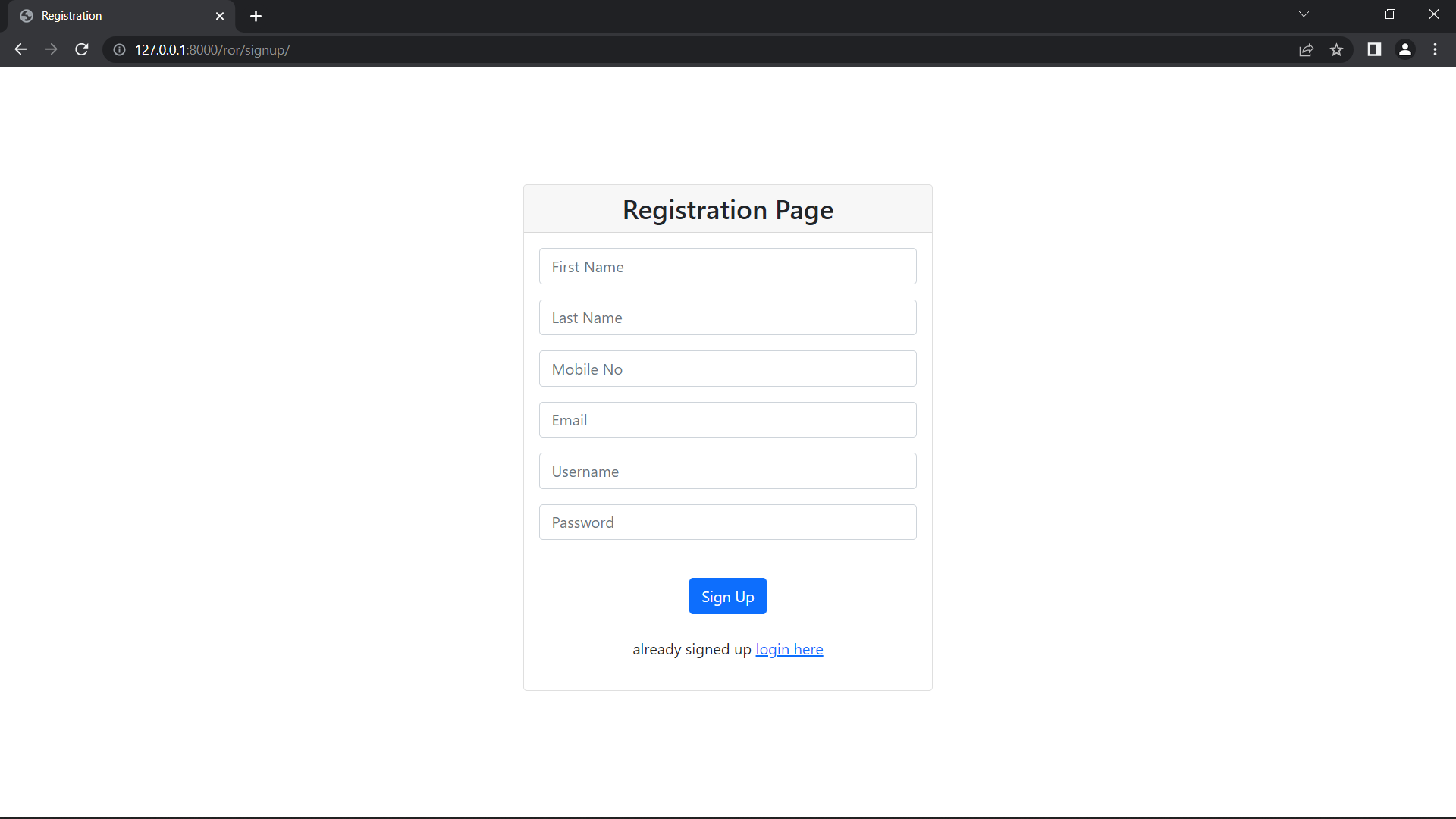
**4.4 Entity Relationship Diagram (ERD)**



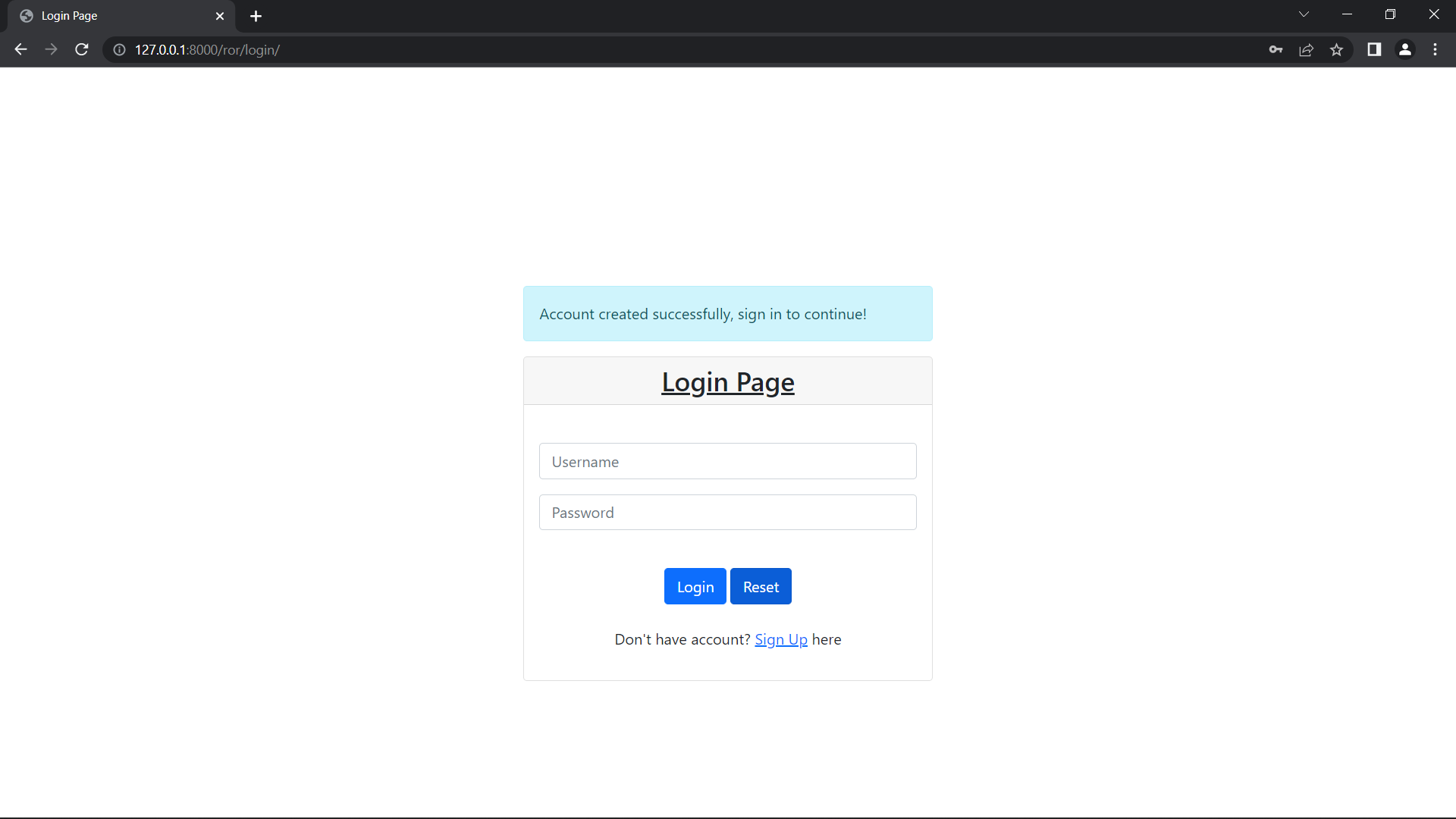
**4.5 Data Dictionary, Table Design, Gantt Chart**

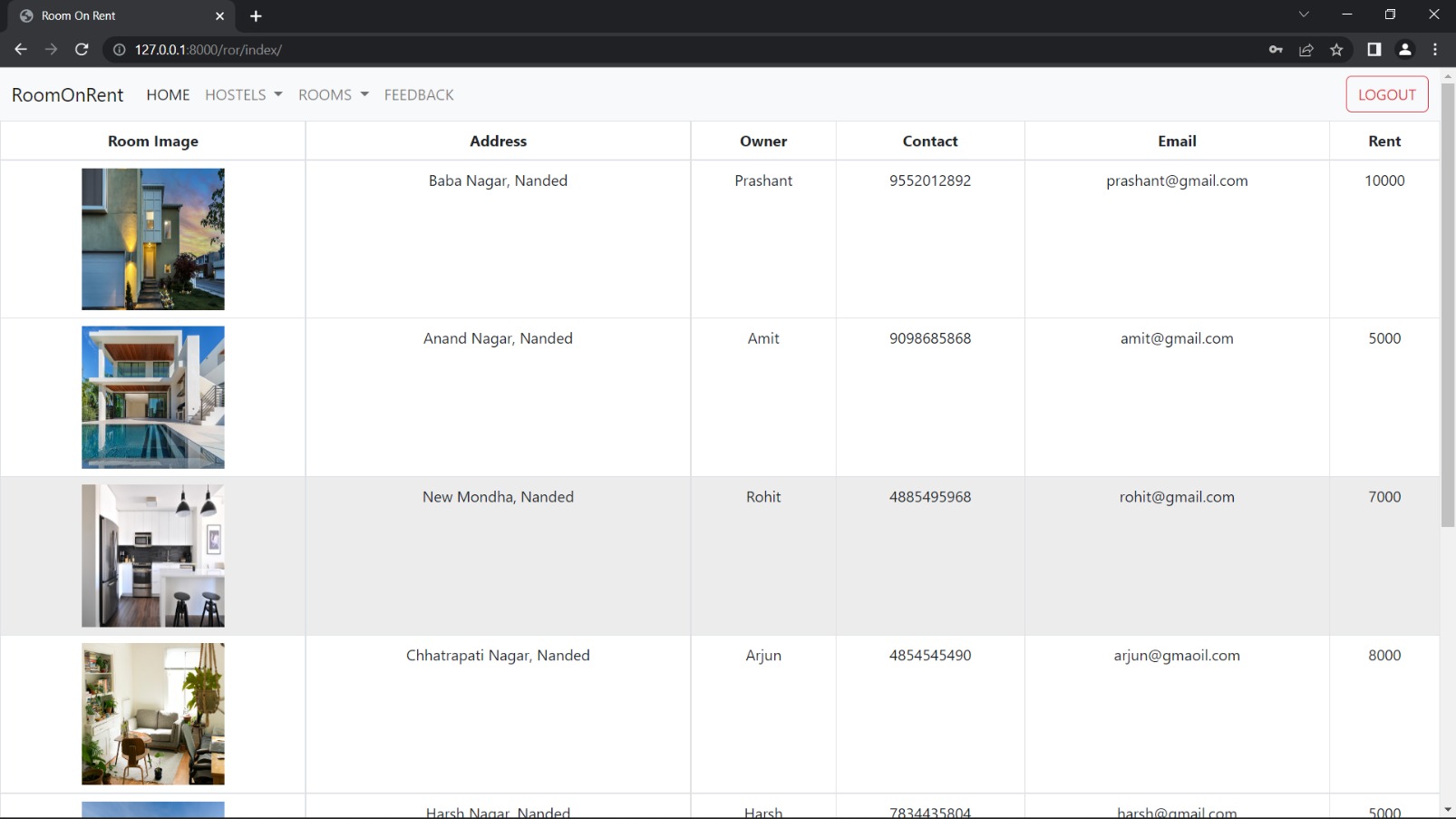
****

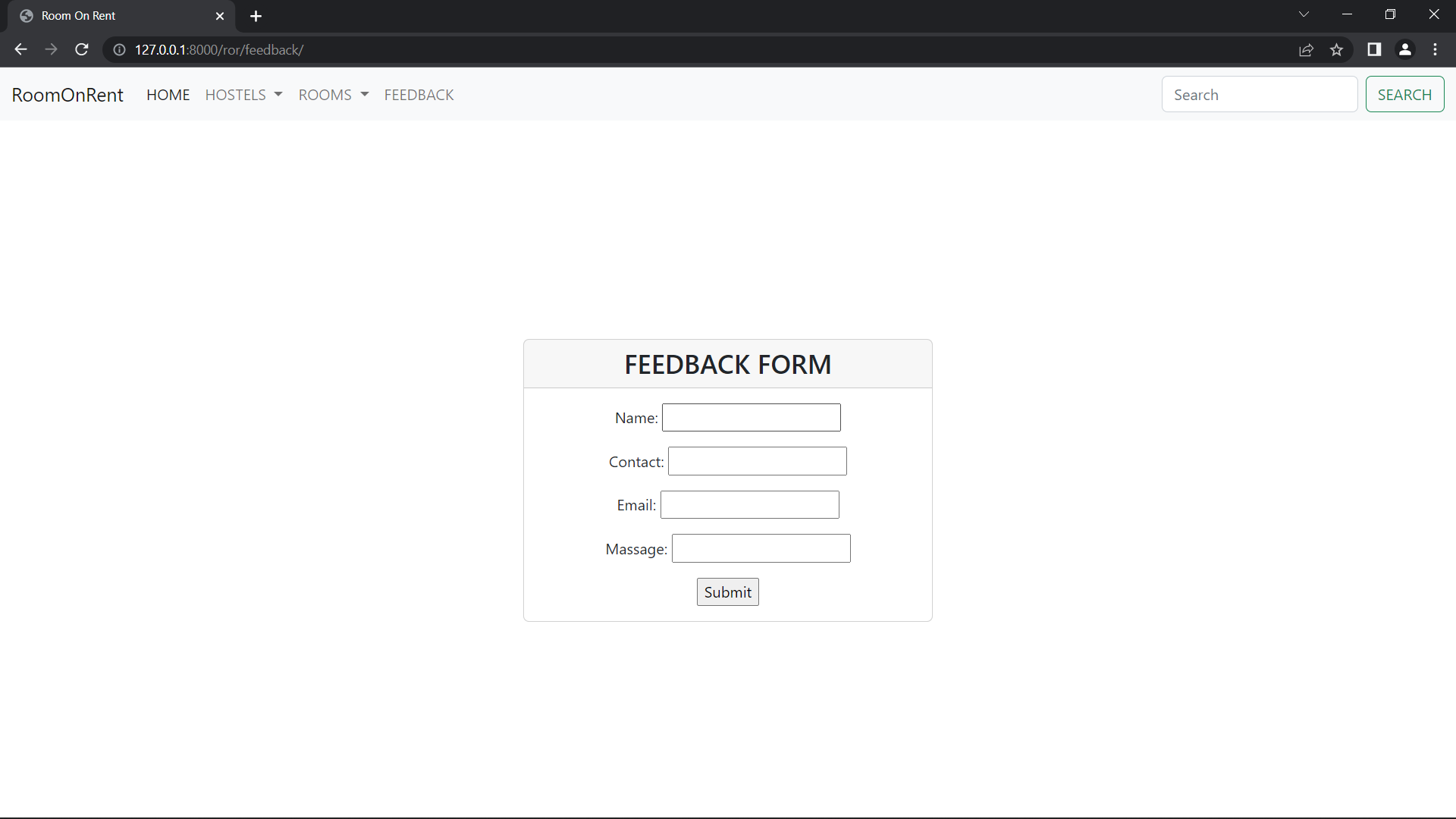
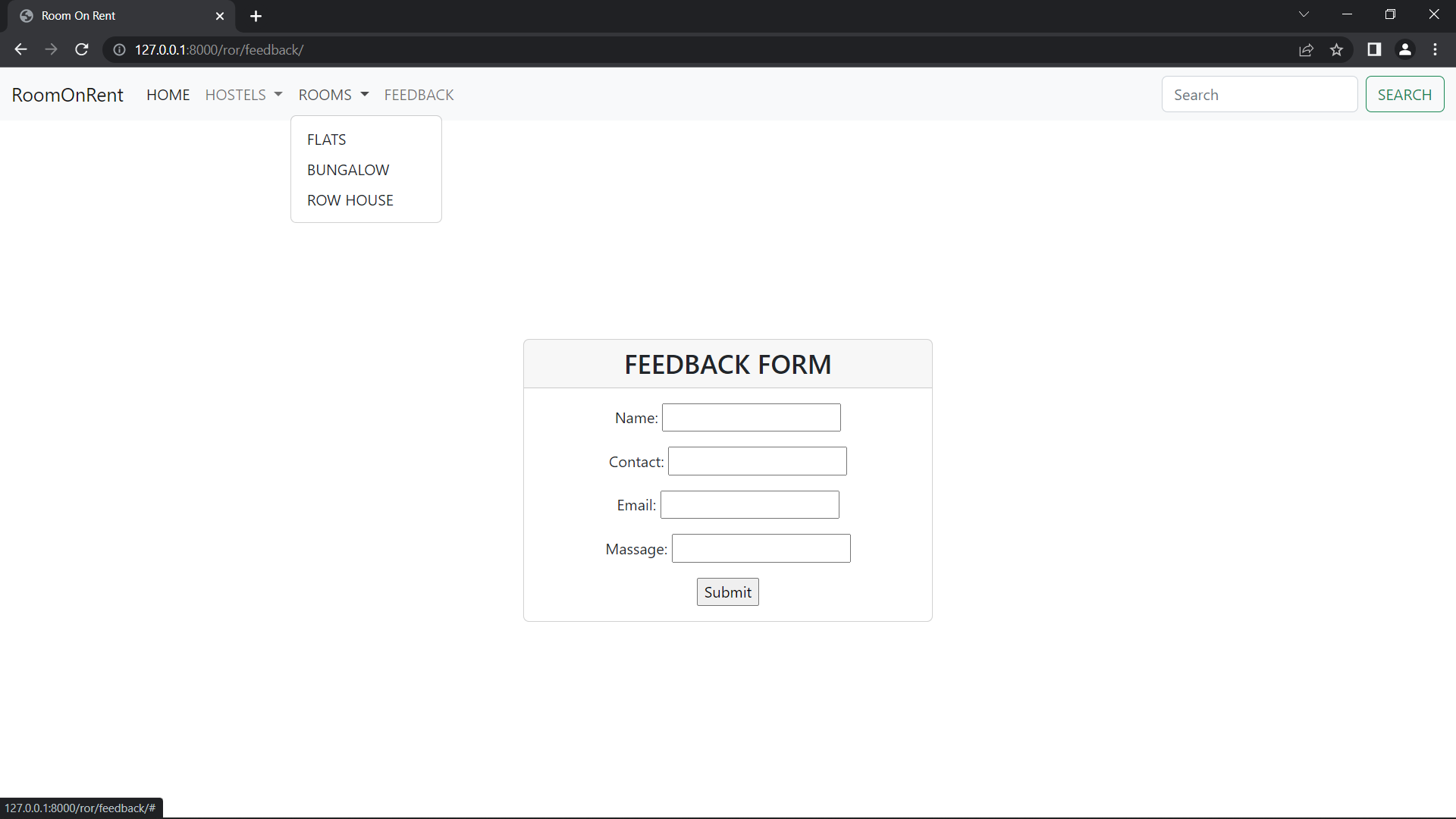
**4.6 Front End Design, Menu Tree, Menu Screens, Input Screens**

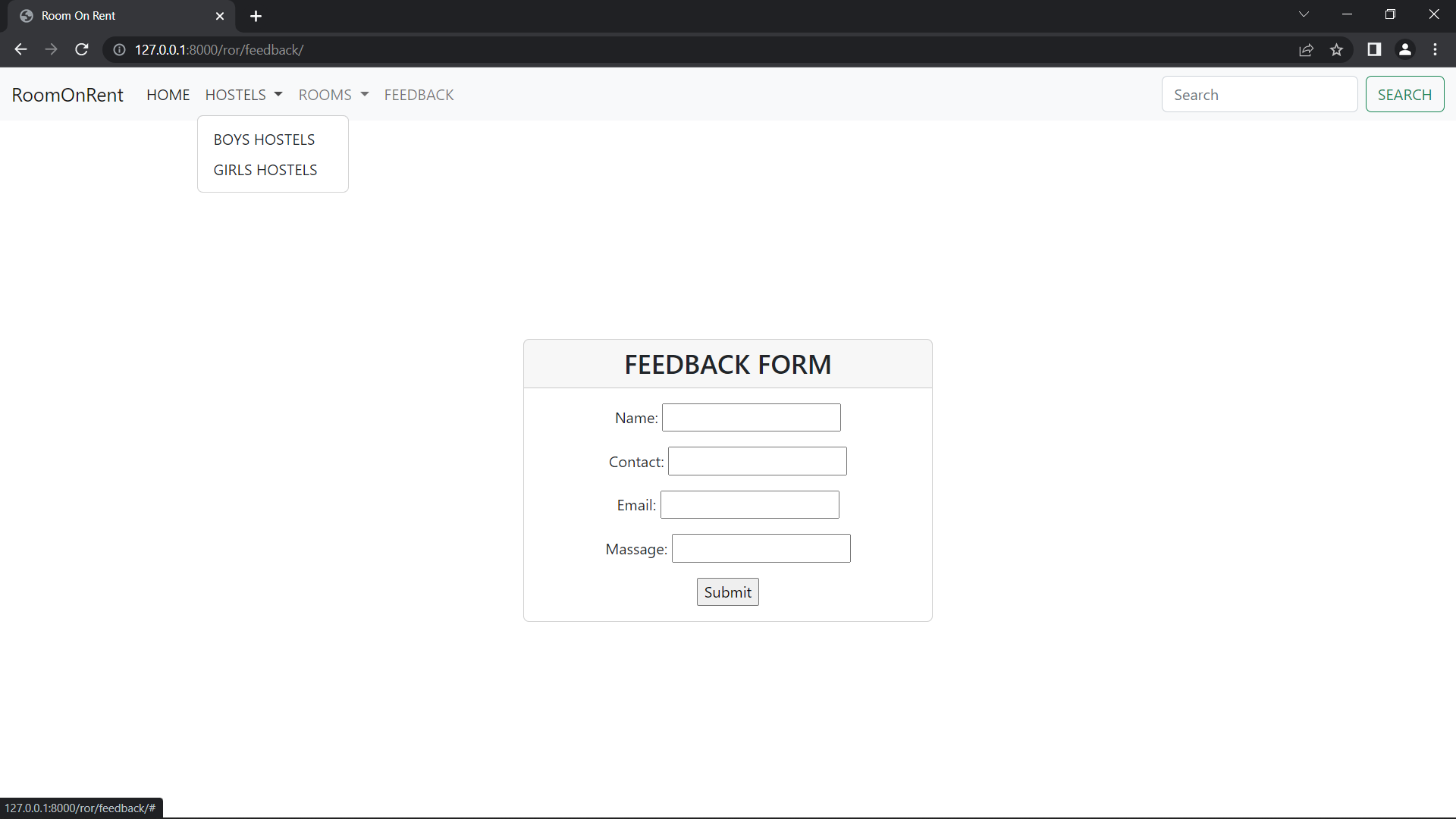
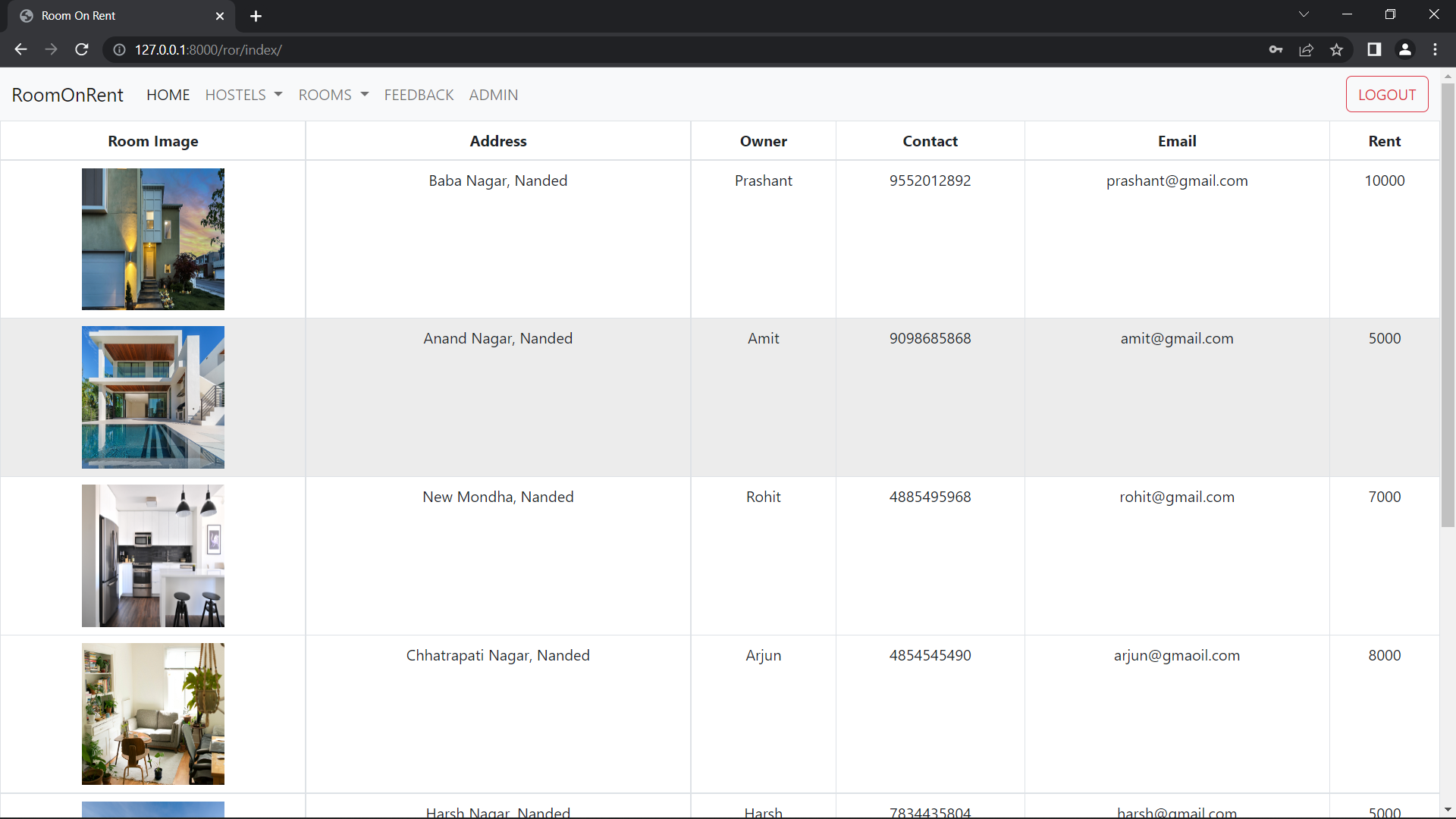


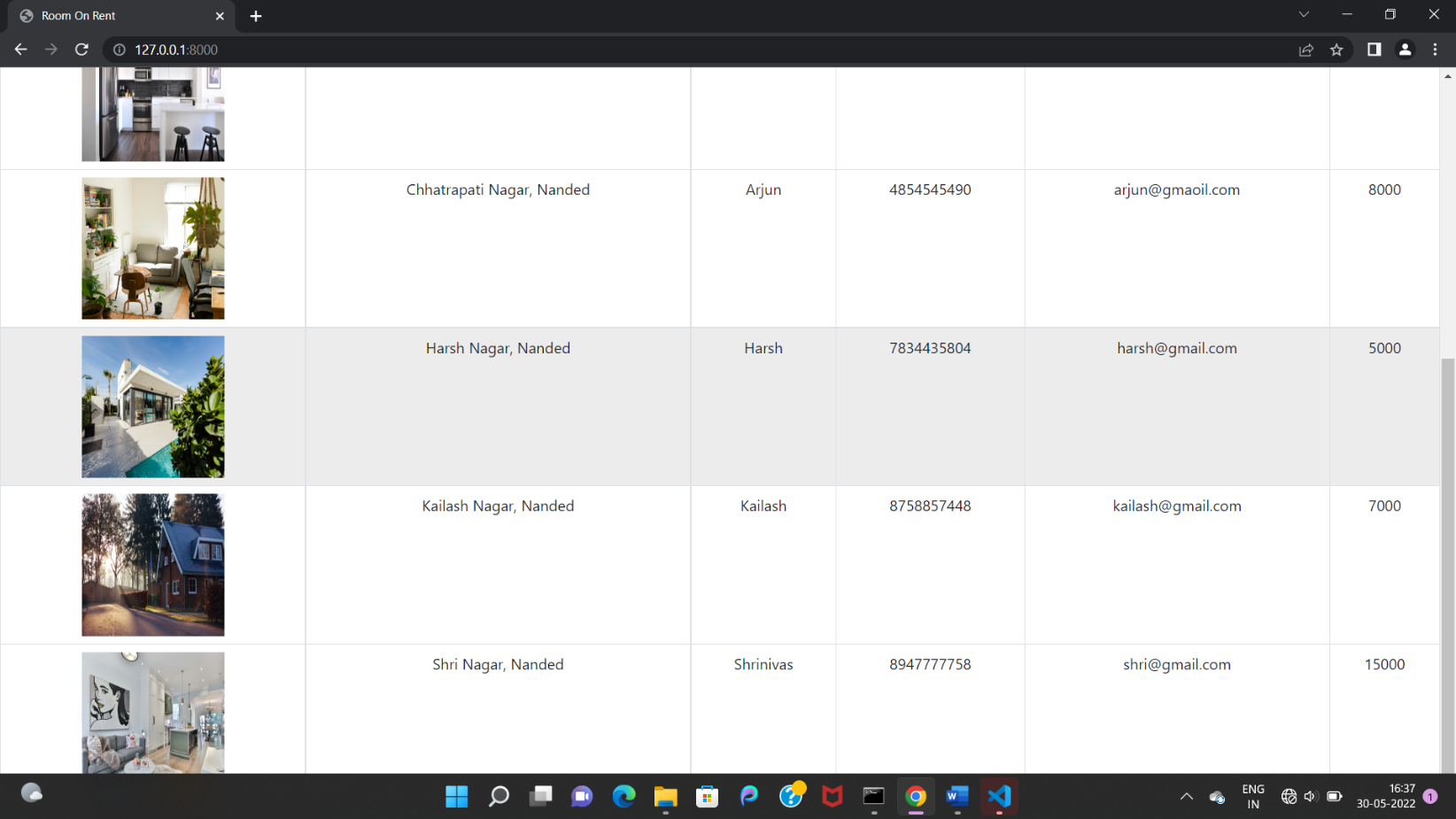
Login page screen

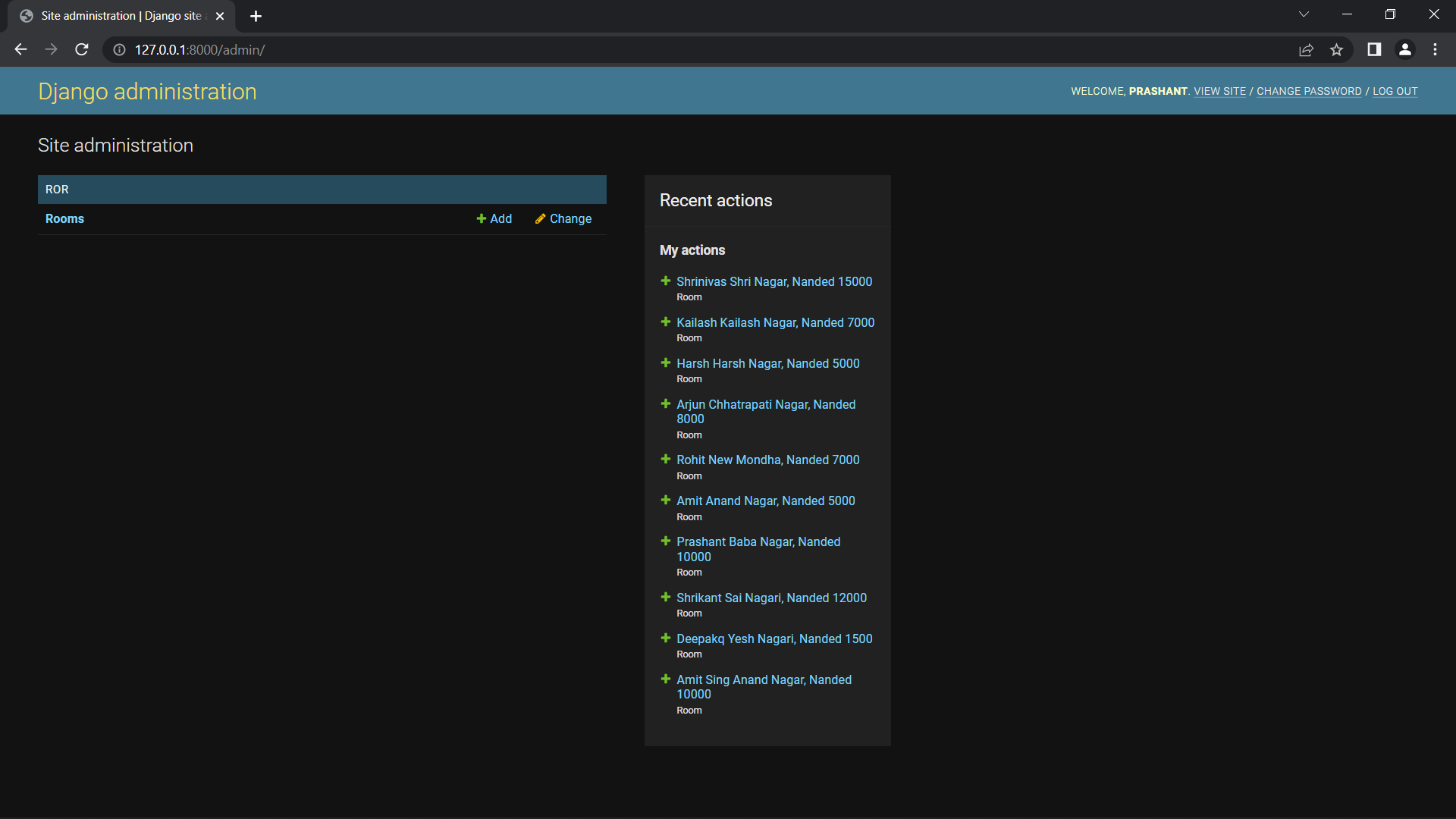


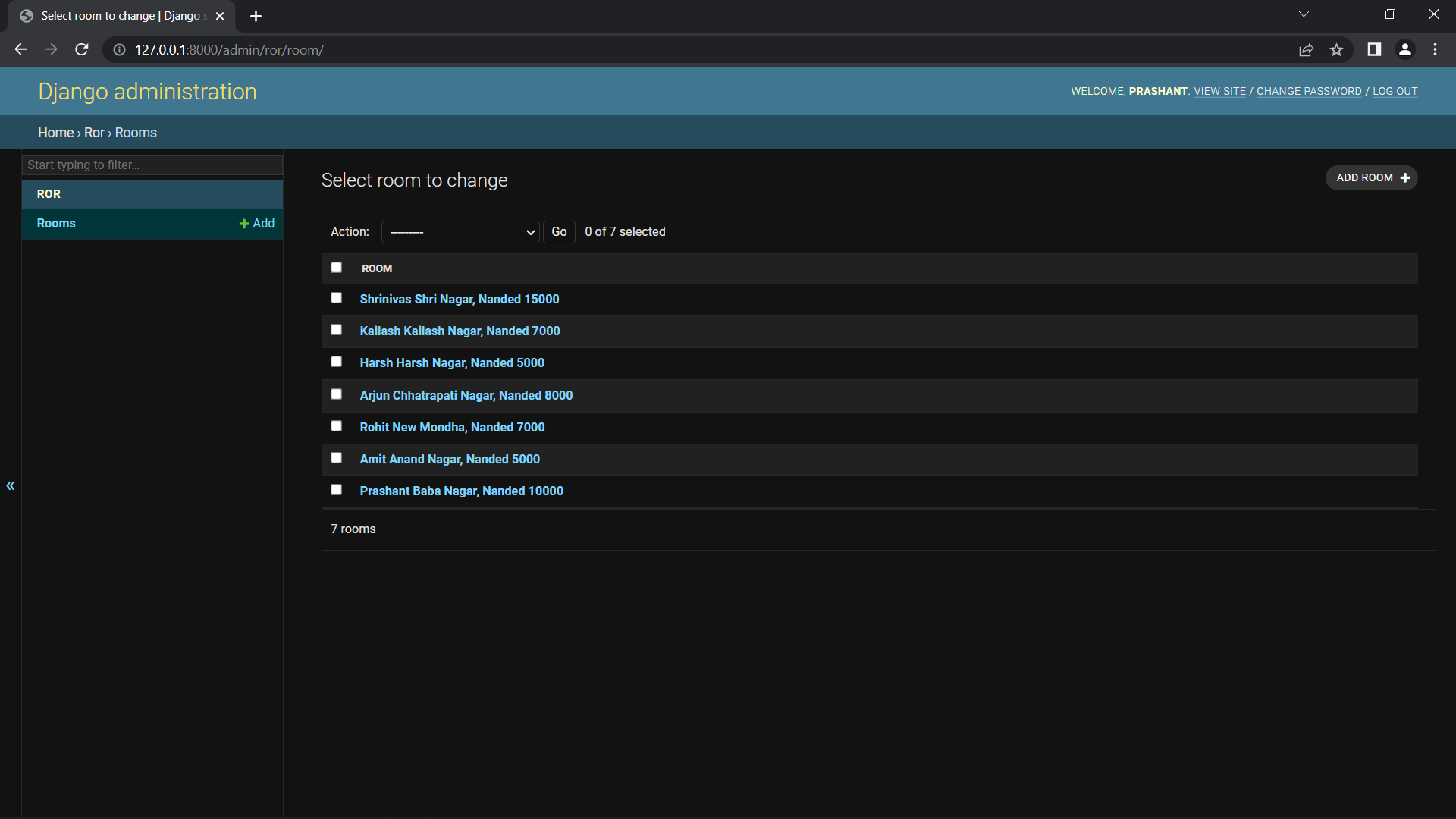
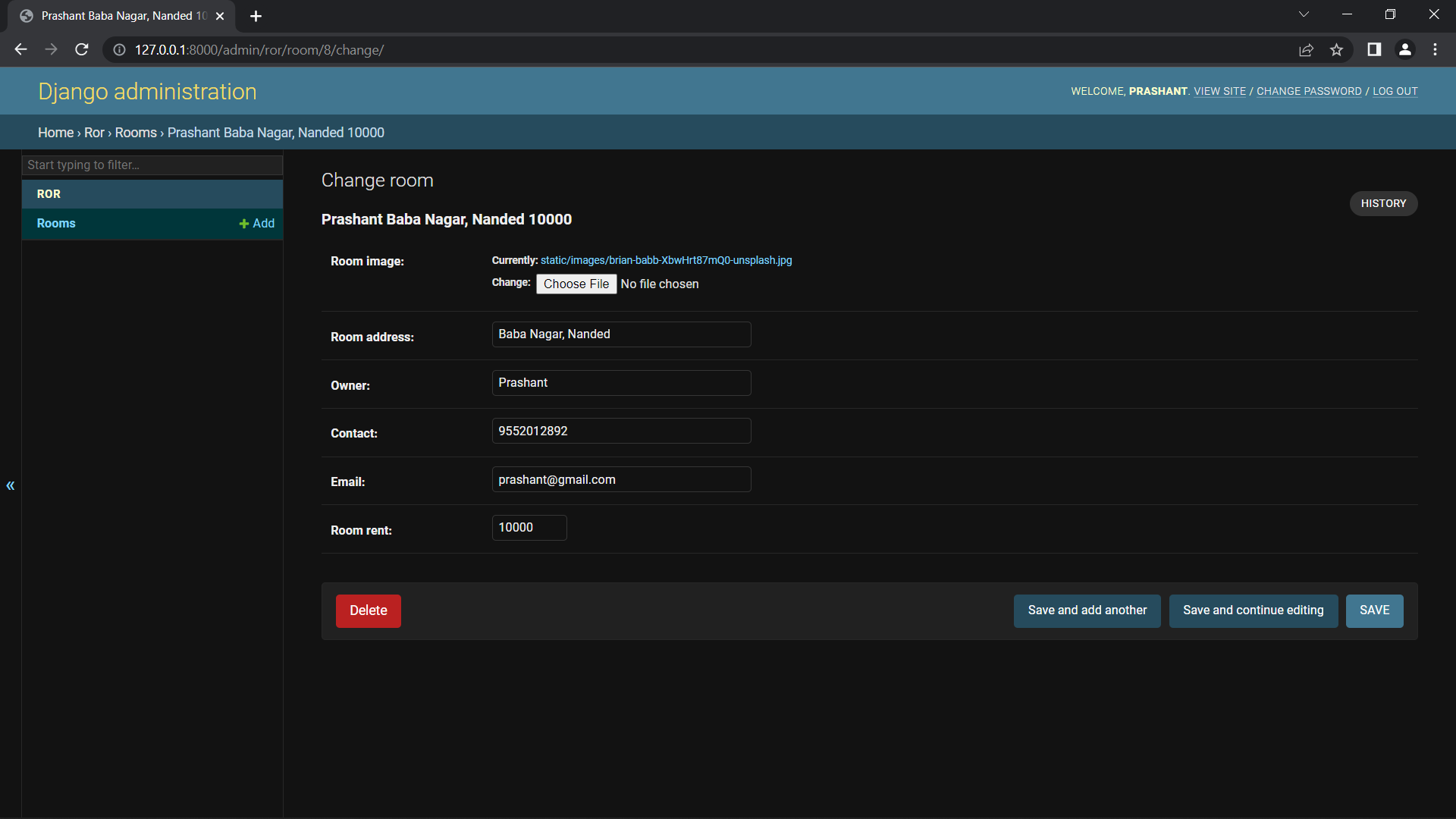


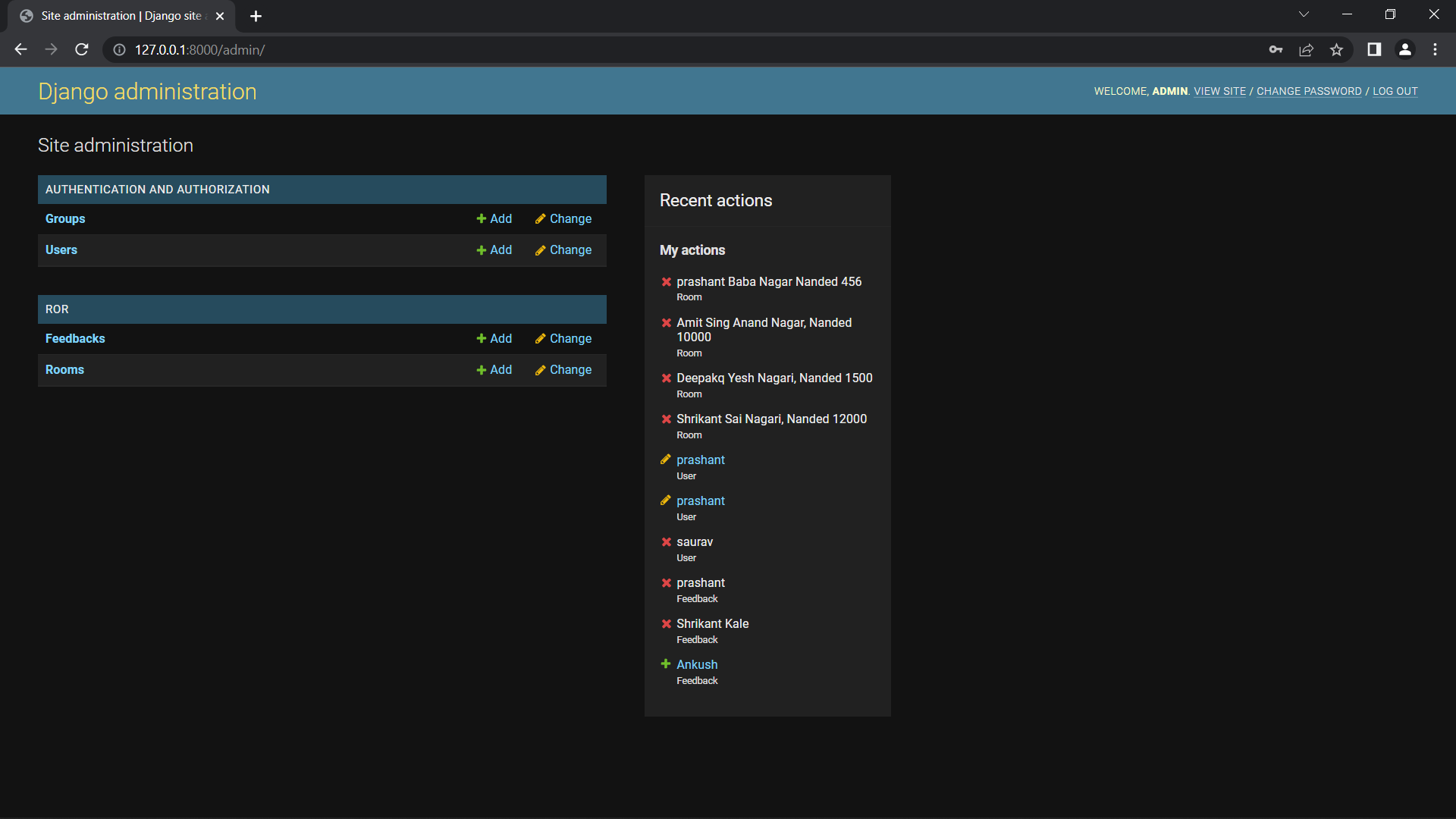
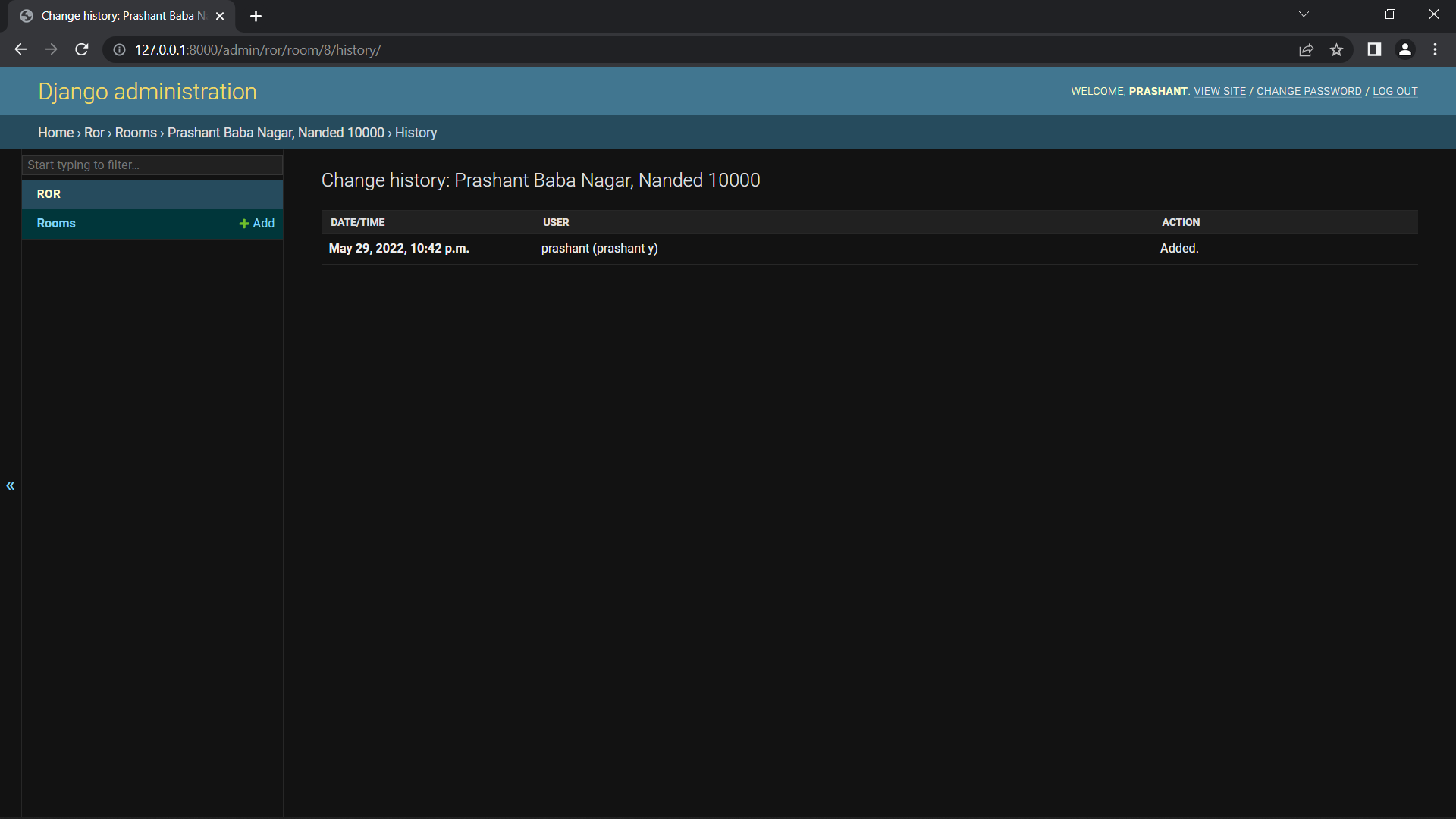
****

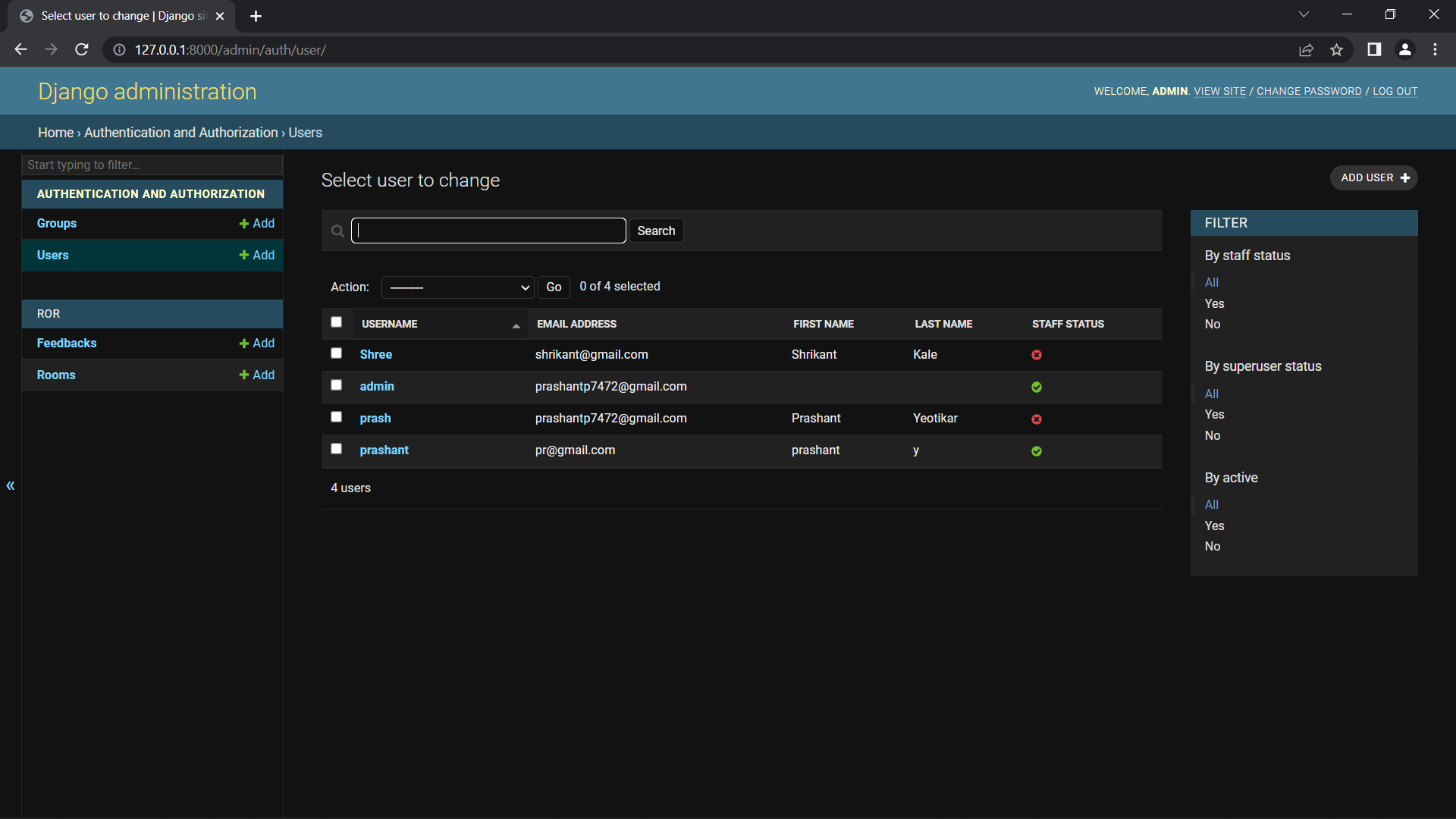
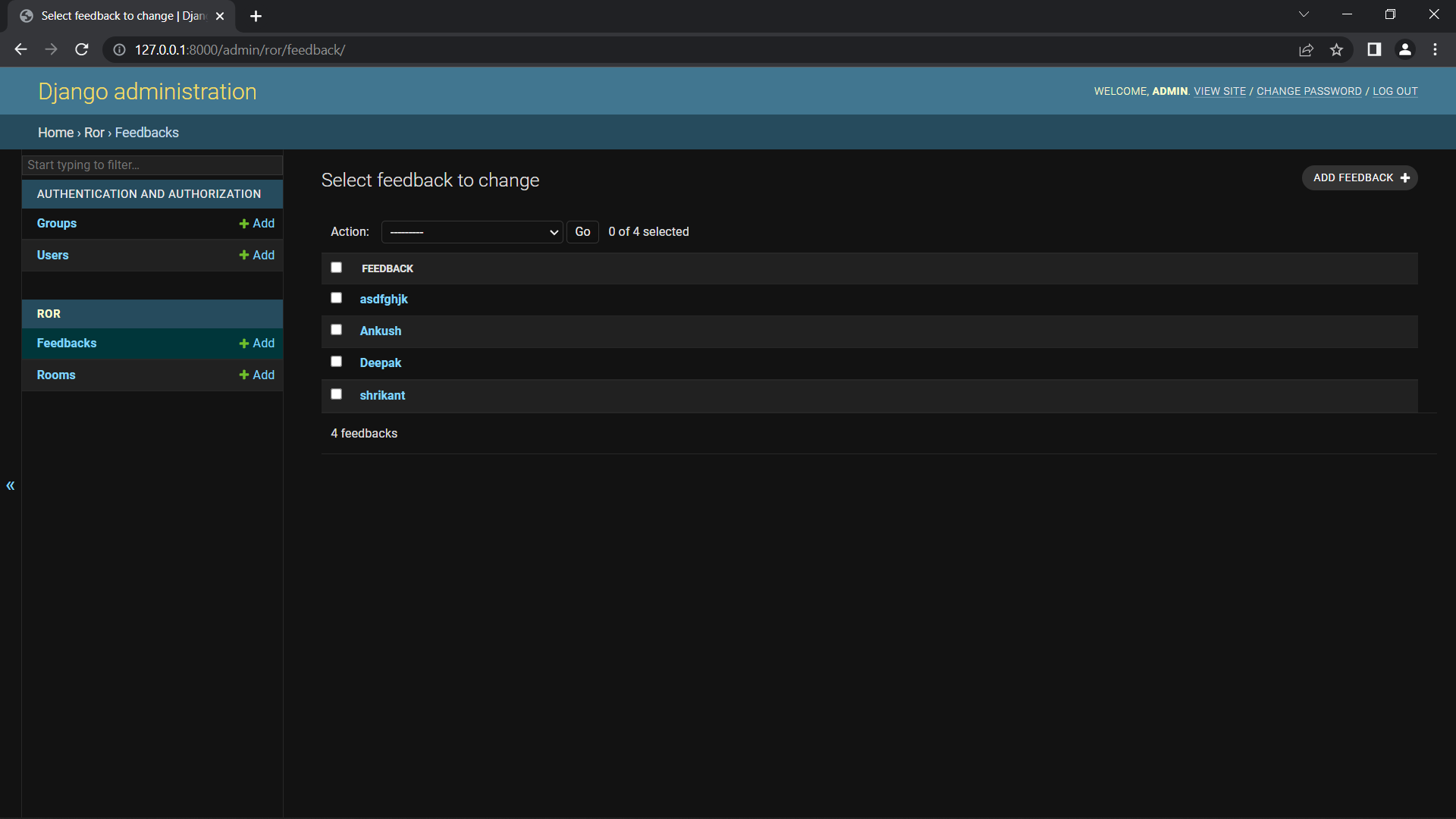
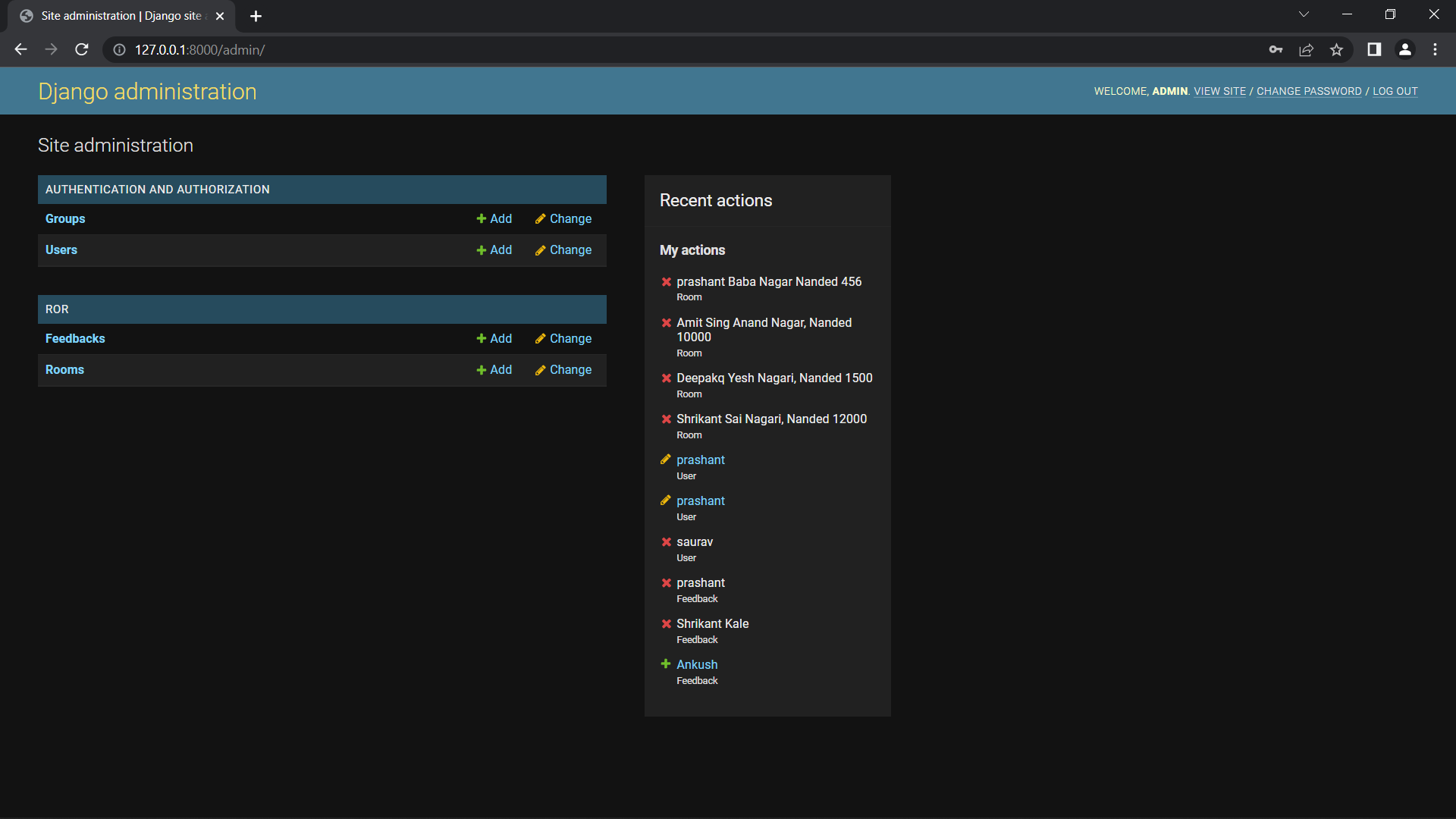
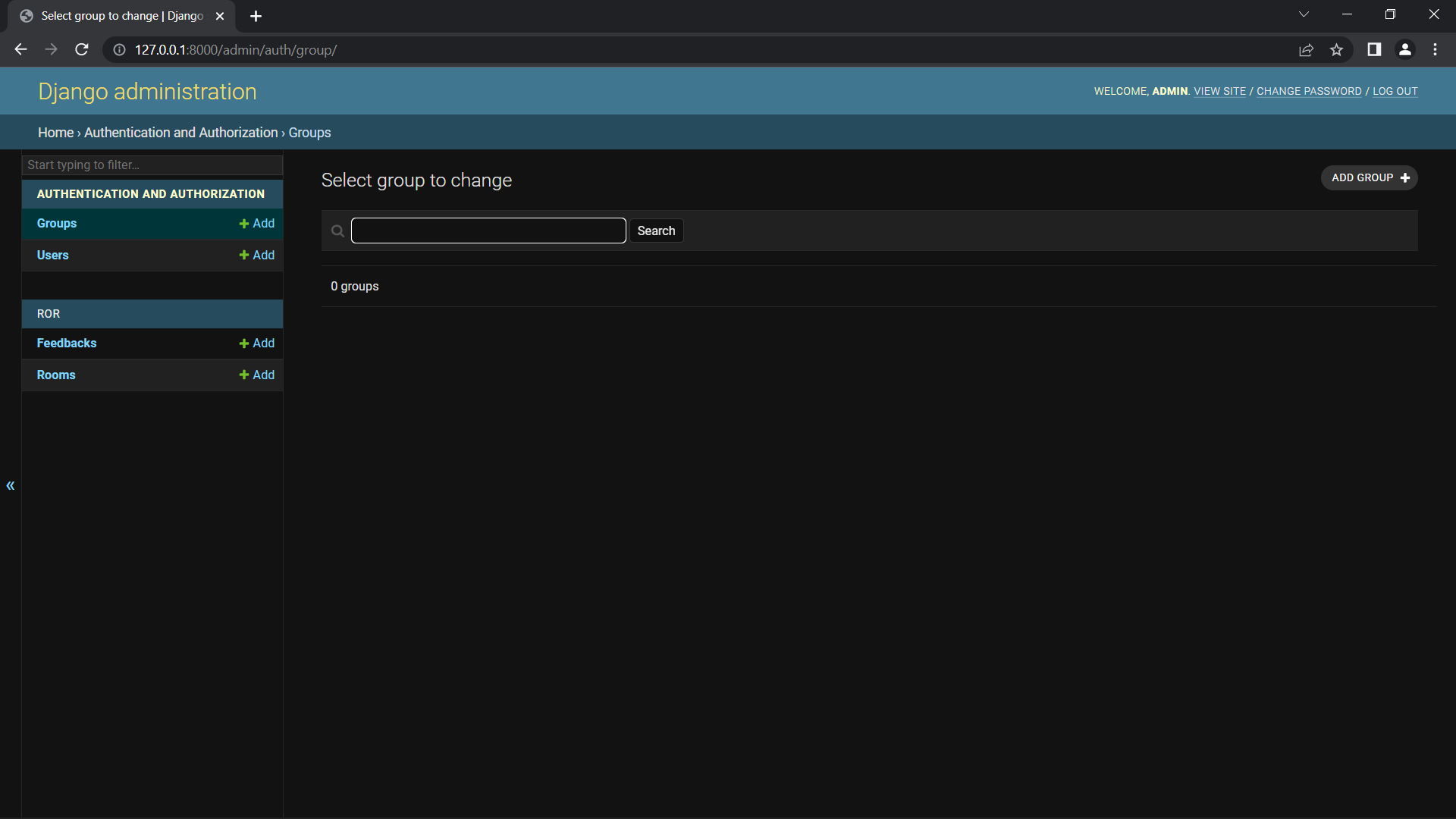
****

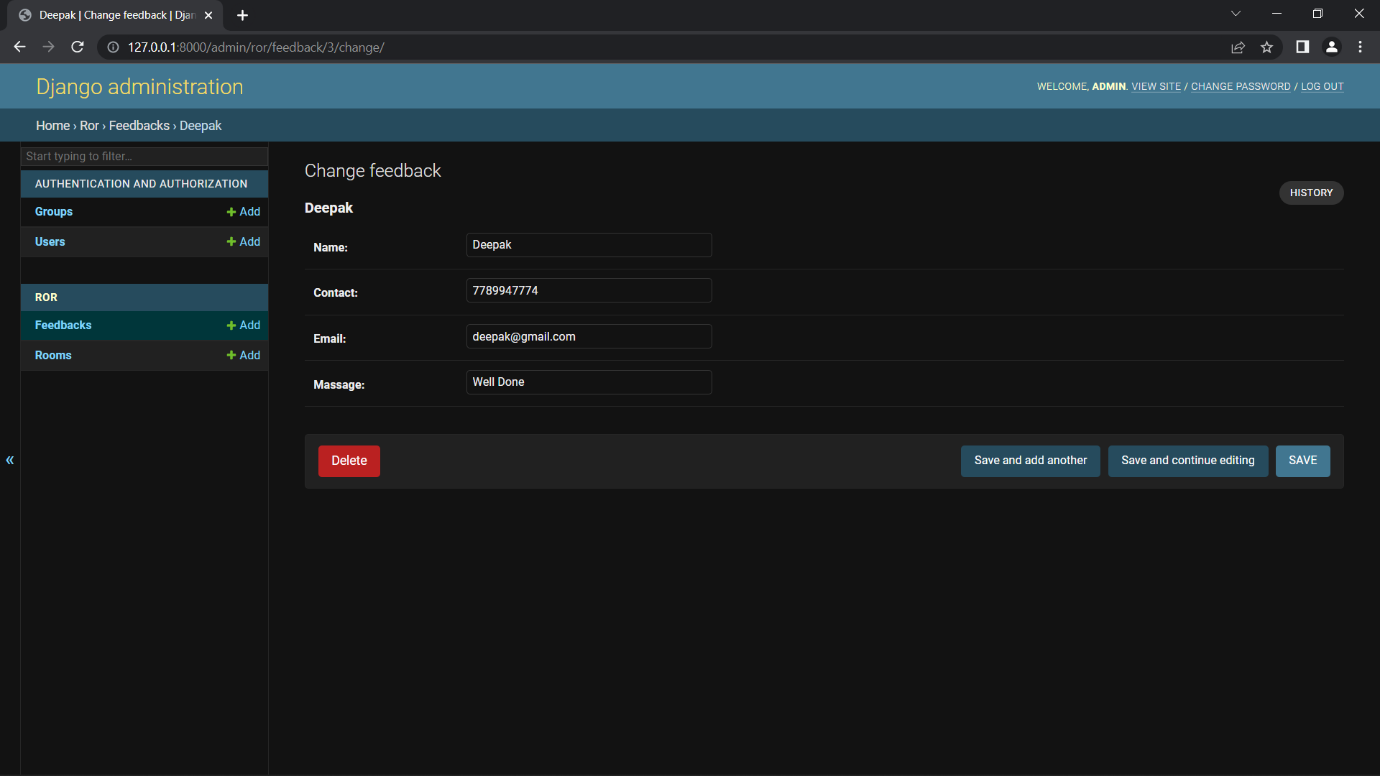
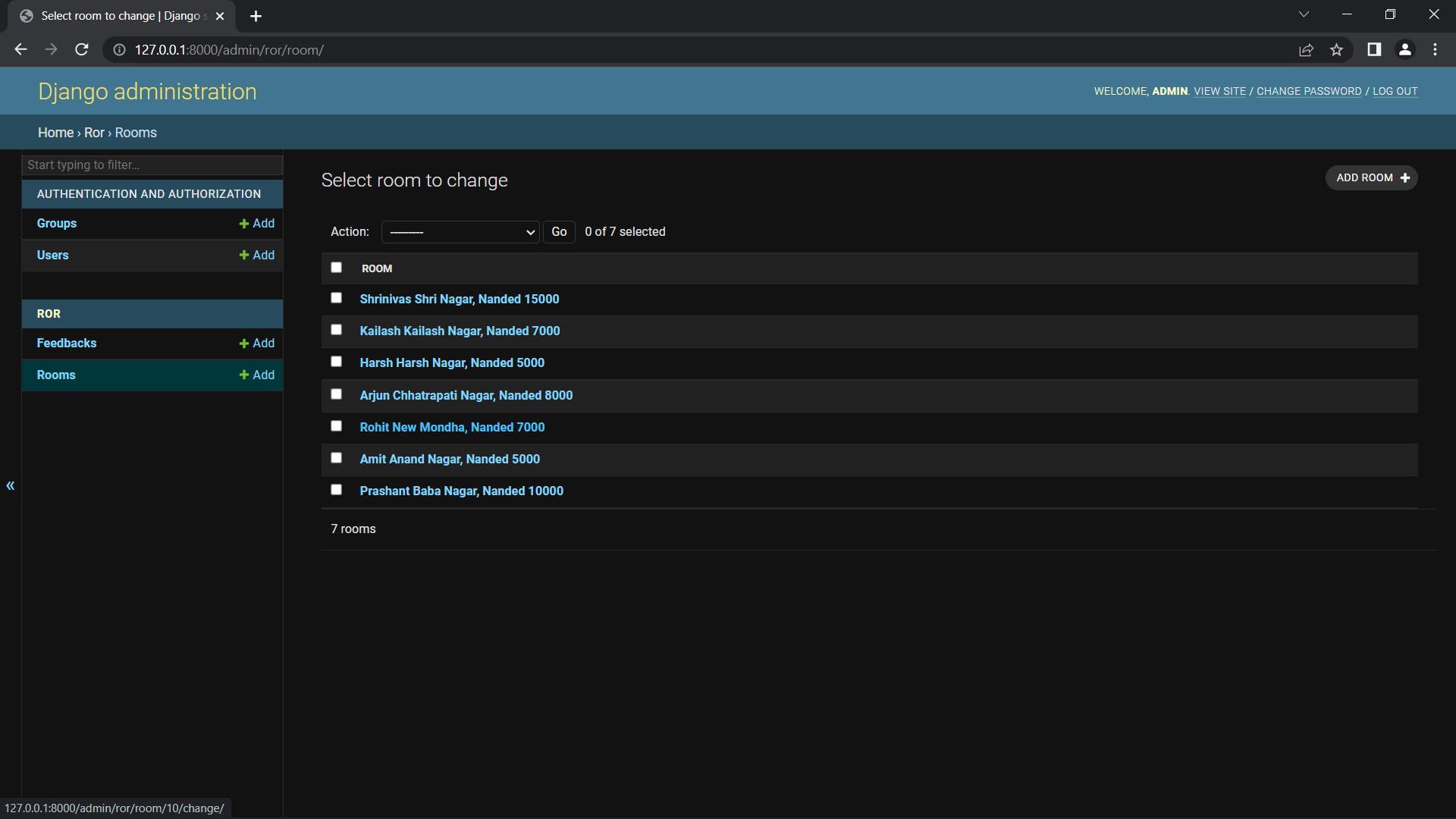
****

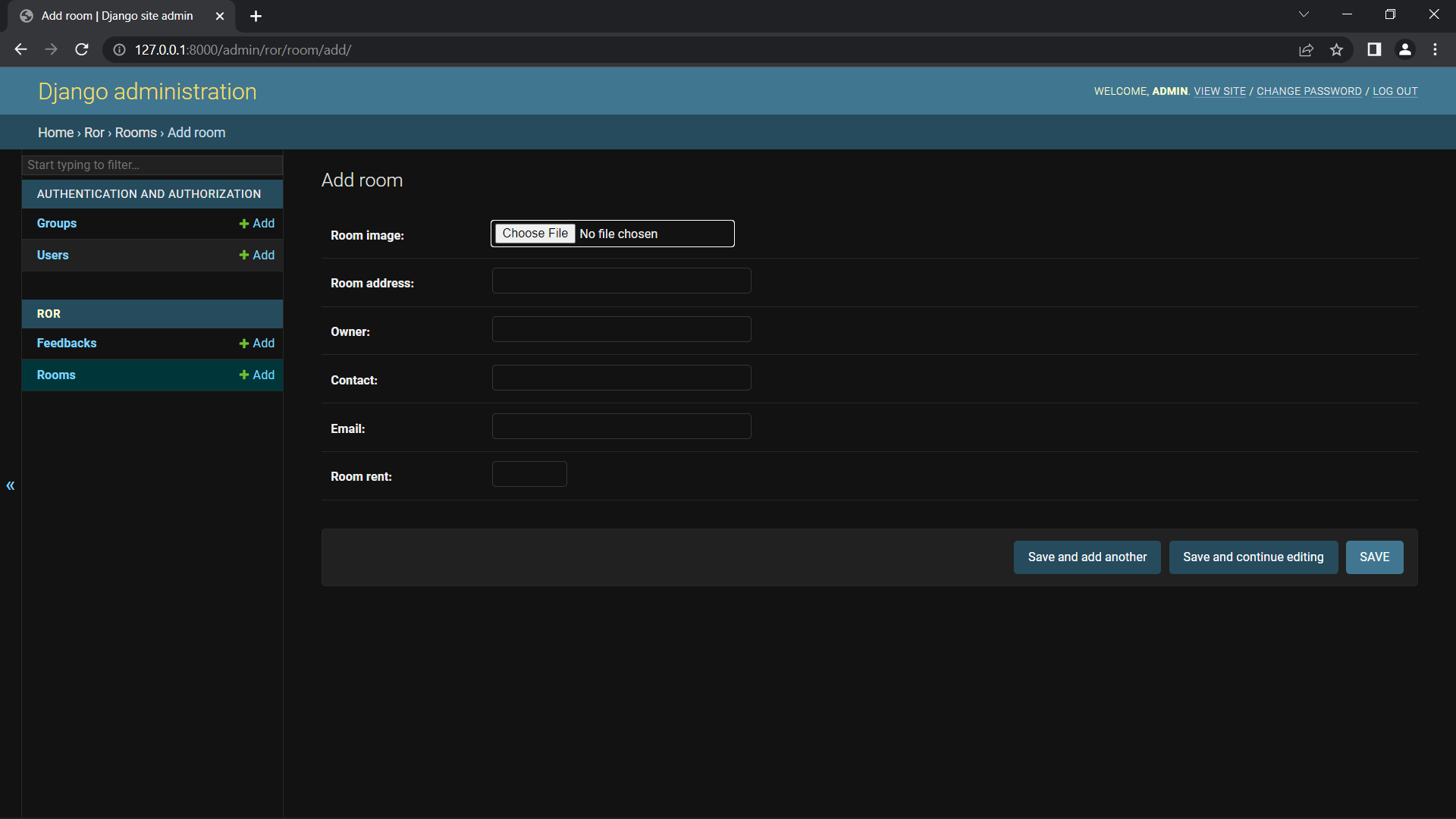
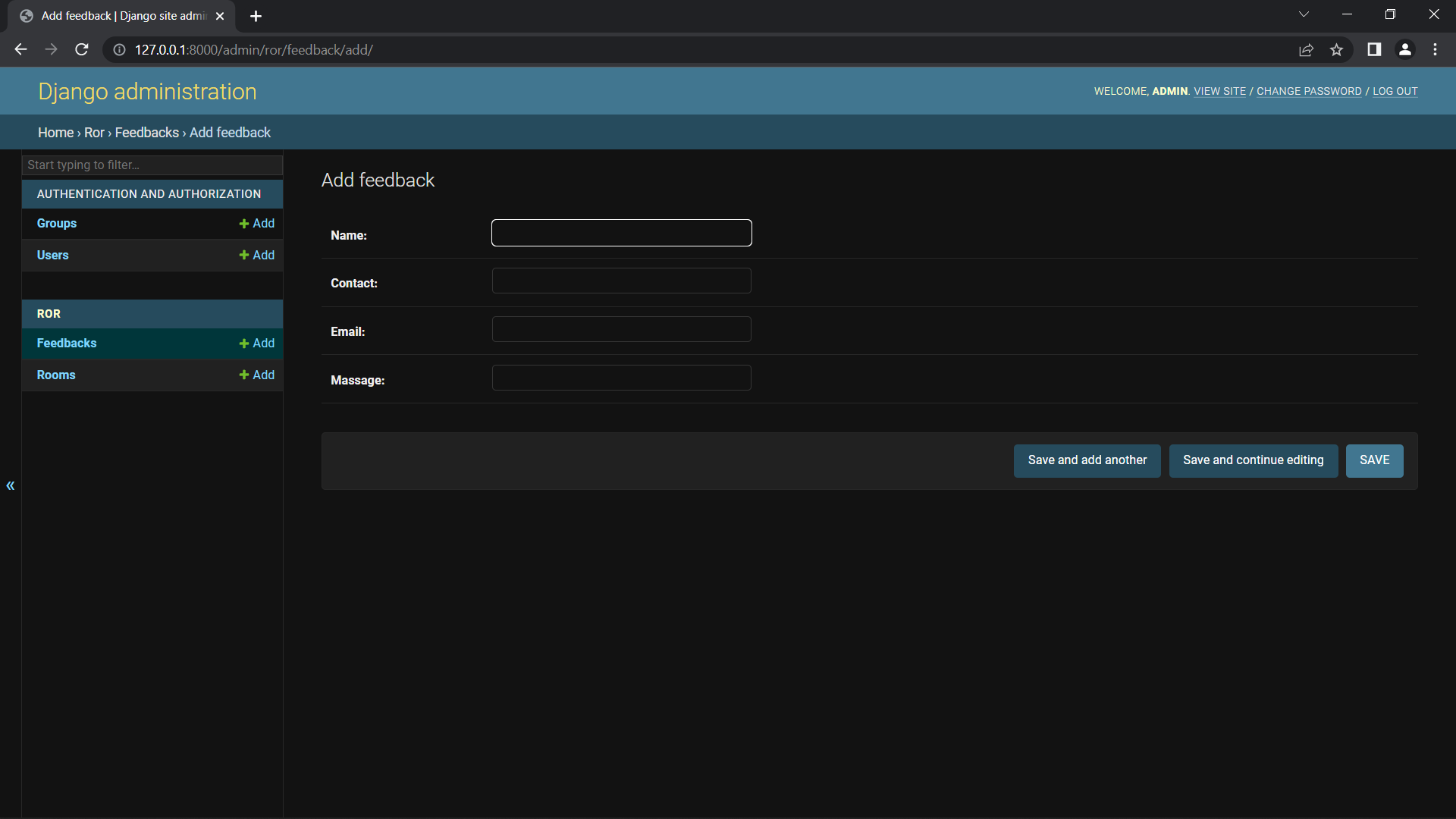
****

****

****

****

****

****

**4.7 Coding**

**INDEX**

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!-- <link rel="stylesheet" href="{% static 'css/main.css' %}"> -->

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css.map' %}">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

  <title>Room On Rent</title>

</head>

<body>

  <nav class="navbar navbar-expand-lg bg-light">

    <div class="container-fluid">

      <a class="navbar-brand" href="#">RoomOnRent</a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0">

          <li class="nav-item">

            <a class="nav-link active" aria-current="page" href="/ror/index/">HOME</a>

          </li>

          <li class="nav-item dropdown">

            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

              HOSTELS

            </a>

            <ul class="dropdown-menu" aria-labelledby="navbarDropdown">

              <li><a class="dropdown-item" href="#">BOYS HOSTELS</a></li>

              <li><a class="dropdown-item" href="#">GIRLS HOSTELS</a></li>

            </ul>

            <li class="nav-item dropdown">

              <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

                ROOMS

              </a>

              <ul class="dropdown-menu" aria-labelledby="navbarDropdown">

                <li><a class="dropdown-item" href="#">FLATS</a></li>

                <li><a class="dropdown-item" href="#">BUNGALOW</a></li>

                <li><a class="dropdown-item" href="#">ROW HOUSE</a></li>

              </ul>

            <li class="nav-item">

              <a class="nav-link" href="/ror/feedback/">FEEDBACK</a>

            </li>

            {% if request.user.is\_staff %}

            <li class="nav-item">

              <a class="nav-link" href="/admin/">ADMIN</a>

            </li>

            {% endif %}

          </li>

        </ul>

        <div class="d-flex" role="logout">

          <a class="btn btn-outline-danger" href="/ror/logout">LOGOUT</a>

        </div>

      </div>

    </div>

  </nav>

  {% if rooms %}

  <table class="table table-hover table-bordered text-center">

    <thead>

      <th>Room Image</th>

      <th>Address</th>

      <th>Owner</th>

      <th>Contact</th>

      <th>Email</th>

      <th>Rent</th>

    </thead>

    <tbody>

      {% for room in rooms %}

        <tr>

          <td>

            <img src="{{ STATIC\_URL }}/{{ room.room\_image }}" alt="room image" width="150px" height="150px">

          </td>

          <td>{{ room.room\_address }}</td>

          <td>{{ room.owner }}</td>

          <td>{{ room.contact }}</td>

          <td>{{ room.email }}</td>

          <td>{{ room.room\_rent }}</td>

        </tr>

      {% endfor %}

    </tbody>

  </table>

  {% endif %}

<script src="{% static 'js/bootstrap.bundle.js' %}"></script>

<script src="{% static 'js/bootstrap.bundle.js.map' %}"></script>

</body>

</html>

**LOGIN PAGE**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"

    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

    <title>Login Page</title>

    <style>

        .container {

            /\* margin: 0 auto;

            float: none;

            margin-bottom: 10px; \*/

            position: relative;

            padding-top: 15%;

        }

    </style>

</head>

<body>

    <div class="container d-flex justify-content-center">

        <div class="col-md-4">

            {% if messages %}

                {% for message in messages %}

                <div role="alert" class="alert alert-info">{{ message }}</div>

                {% endfor %}

            {% endif %}

            <div class="card">

                <div class="card-header h3 text-center">

                    <u>Login Page</u>

                </div>

                <div class="card-body text-center">

                    <form method="post">

                        {% csrf\_token %}

                        <div class="mb-3">

                            <!-- <label for="exampleInputUsername" class="form-label">Email address</label> -->

                            <br>

                            <input type="text" class="form-control" name="username" placeholder="Username"

                                id="inputUsername" aria-describedby="emailHelp" required>

                            <!-- <div id="emailHelp" class="form-text">usernames are case sensitive!</div> -->

                        </div>

                        <div class="mb-3">

                            <!-- <label for="exampleInputPassword1" class="form-label">Password</label> -->

                            <input type="password" placeholder="Password" name="password" class="form-control"

                                id="inputPassword" required>

                        </div>

                        <br>

                        <button type="submit" class="btn btn-primary">Login</button>

                        <button type="reset" class="btn btn-primary">Reset</button>

                    </form>

                    <br>

                    <p>Don't have account? <a href='/ror/signup'>Sign Up</a> here</p>

                </div>

            </div>

        </div>

    </div>

    </div>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"

        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"

        crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->

    <!--

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF" crossorigin="anonymous"></script>

-->

</body>

</html>

**SIGNUP PAGE**

<!doctype html>

<html lang="en">

    <head>

        <!-- Required meta tags -->

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-scale=1">

        <!-- Bootstrap CSS -->

        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"

            integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

        <title>Registration</title>

        <style>

            .container {

                /\* margin: 0 auto;

                float: none;

                margin-bottom: 10px; \*/

                position: relative;

                padding-top: 8%;

            }

        </style>

    </head>

    <body>

        <div class="container d-flex justify-content-center">

            <div class="col-md-4">

                {% if messages %}

                    {% for message in messages %}

                    <div role="alert"

                        class="alert alert-info ">{{ message }}</div>

                    {% endfor %}

                {% endif %}

                <div class="card">

                    <div class="card-header h3 text-center">

                        Registration Page

                    </div>

                    <div class="card-body text-center">

                        <form method="POST">

                            {% csrf\_token %}

                            <div class="mb-3">

                                <input type="text" class="form-control" name="first\_name" placeholder="First Name" required>

                            </div>

                            <div class="mb-3">

                                <input type="text" class="form-control" name="last\_name" placeholder="Last Name">

                            </div>

                            <div class="mb-3">

                                <input type="mobno" class="form-control" name="mobno" placeholder="Mobile No" id="mobno" required>

                            </div>

                            <div class="mb-3">

                                <input type="email" class="form-control" name="email" placeholder="Email" id="email" required>

                            </div>

                            <div class="mb-3">

                                <input type="text" class="form-control" name="username" placeholder="Username"

                                    id="Username" required>

                            </div>

                            <div class="mb-3">

                                <input type="password" placeholder="Password" name="password" class="form-control"

                                    id="Password1" required>

                            </div>

                            <br>

                            <!--                    <div class="mb-3">-->

                            <!--                        <input type="password" placeholder="Confirm Password" name="confirm\_password"-->

                            <!--                               class="form-control"-->

                            <!--                               id="confirm\_pass">-->

                            <!--                    </div>-->

                            <button type="submit" class="btn btn-primary">Sign Up</button>

                        </form>

                        <br>

                        <p>already signed up <a href="/ror/login/">login here</a> </p>

                    </div>

                </div>

            </div>

        </div>

        </div>

        <!-- Optional JavaScript; choose one of the two! -->

        <!-- Option 1: Bootstrap Bundle with Popper -->

        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"

            integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"

            crossorigin="anonymous"></script>

        <!-- Option 2: Separate Popper and Bootstrap JS -->

        <!--

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF" crossorigin="anonymous"></script>

-->

    </body>

</html>

**ADD ROOMS PAGE**

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!-- <link rel="stylesheet" href="{% static 'css/main.css' %}"> -->

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css.map' %}">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

  <title>Room On Rent</title>

</head>

<body>

  <nav class="navbar navbar-expand-lg bg-light">

    <div class="container-fluid">

      <a class="navbar-brand" href="#">RoomOnRent</a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0">

          <li class="nav-item">

            <a class="nav-link active" aria-current="page" href="/ror/index/">HOME</a>

          </li>

          <li class="nav-item dropdown">

            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

              HOSTELS

            </a>

            <ul class="dropdown-menu" aria-labelledby="navbarDropdown">

              <li><a class="dropdown-item" href="#">BOYS HOSTELS</a></li>

              <li><a class="dropdown-item" href="#">GIRLS HOSTELS</a></li>

            </ul>

            <li class="nav-item dropdown">

              <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

                ROOMS

              </a>

              <ul class="dropdown-menu" aria-labelledby="navbarDropdown">

                <li><a class="dropdown-item" href="#">FLATS</a></li>

                <li><a class="dropdown-item" href="#">BUNGALOW</a></li>

                <li><a class="dropdown-item" href="#">ROW HOUSE</a></li>

              </ul>

            <li class="nav-item">

              <a class="nav-link" href="#">FEEDBACK</a>

            </li>

          </li>

        </ul>

        <!--

        <form class="d-flex" role="search">

          <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

          <button class="btn btn-outline-success" type="submit">SEARCH</button>

        </form>

        -->

      </div>

    </div>

  </nav>

  <form method="post">

    {% csrf\_token %}

    {{ form.as\_p }}

    <input type="submit" value="Submit">

</form>

<script src="{% static 'js/bootstrap.bundle.js' %}"></script>

<script src="{% static 'js/bootstrap.bundle.js.map' %}"></script>

</body>

</html>

**FEEDBACK PAGE**

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!-- <link rel="stylesheet" href="{% static 'css/main.css' %}"> -->

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css.map' %}">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

  <title>Room On Rent</title>

  <style>

        .container {

            /\* margin: 0 auto;

            float: none;

            margin-bottom: 10px; \*/

            position: relative;

            padding-top: 15%;

        }

    form {text-align: center;}

  </style>

</head>

<body>

  <nav class="navbar navbar-expand-lg bg-light">

    <div class="container-fluid">

      <a class="navbar-brand" href="#">RoomOnRent</a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0">

          <li class="nav-item">

            <a class="nav-link active" aria-current="page" href="/ror/index/">HOME</a>

          </li>

          <li class="nav-item dropdown">

            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

              HOSTELS

            </a>

            <ul class="dropdown-menu" aria-labelledby="navbarDropdown">

              <li><a class="dropdown-item" href="#">BOYS HOSTELS</a></li>

              <li><a class="dropdown-item" href="#">GIRLS HOSTELS</a></li>

            </ul>

            <li class="nav-item dropdown">

              <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

                ROOMS

              </a>

              <ul class="dropdown-menu" aria-labelledby="navbarDropdown">

                <li><a class="dropdown-item" href="#">FLATS</a></li>

                <li><a class="dropdown-item" href="#">BUNGALOW</a></li>

                <li><a class="dropdown-item" href="#">ROW HOUSE</a></li>

              </ul>

            <li class="nav-item">

              <a class="nav-link" href="/ror/feedback/">FEEDBACK</a>

            </li>

          </li>

          {% if request.user.is\_staff %}

          <li class="nav-item">

            <a class="nav-link" href="/admin/">ADMIN</a>

          </li>

          {% endif %}

        </ul>

        <form class="d-flex" role="search">

          <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

          <button class="btn btn-outline-success" type="submit">SEARCH</button>

        </form>

      </div>

    </div>

  </nav>

  <div class="container d-flex justify-content-center">

    <div class="col-md-4">

        {% if messages %}

            {% for message in messages %}

            <div role="alert" class="alert alert-info">{{ message }}</div>

            {% endfor %}

        {% endif %}

        <div class="card">

            <div class="card-header h3 text-center">

                FEEDBACK FORM

            </div>

            <div class="card-body text-center">

                <form method="post">

                   {% csrf\_token %}

                   {{ form.as\_p }}

                   <input type="Submit" value="Submit">

                   </form>

            </div>

        </div>

    </div>

</div>

</div>

  <!-- start your logic from here -->

<script src="{% static 'js/bootstrap.bundle.js' %}"></script>

<script src="{% static 'js/bootstrap.bundle.js.map' %}"></script>

</body>

</html>

**CSS**

/\* Search Bar \*/

body {

    font-family: Arial;

  }

  \* {

    box-sizing: border-box;

  }

  form.example input[type=text] {

    padding: 10px;

    font-size: 17px;

    border: 1px solid grey;

    float: left;

    width: 80%;

    background: #f1f1f1;

  }

  form.example button {

    float: left;

    width: 20%;

    padding: 10px;

    background: #2196F3;

    color: white;

    font-size: 17px;

    border: 1px solid grey;

    border-left: none;

    cursor: pointer;

  }

  form.example button:hover {

    background: #0b7dda;

  }

  form.example::after {

    content: "";

    clear: both;

    display: table;

  }

**ASGI.PY**

"""

ASGI config for roomonrent project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/4.0/howto/deployment/asgi/

"""

**IMPORT OS**

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'roomonrent.settings')

application = get\_asgi\_application()

**SETTINGS.PY**

"""

Django settings for roomonrent project.

Generated by 'django-admin startproject' using Django 4.0.4.

For more information on this file, see

https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/4.0/ref/settings/

"""

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-@cve)qig2pk@07wiq^-g6)jt3bfo^)k)mj60t7&$k2w+ba6slm'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'ror',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'roomonrent.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [BASE\_DIR / 'templates'],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'roomonrent.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# Password validation

# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/Kolkata'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC\_URL = 'static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static',

            BASE\_DIR / 'room\_images',

]

# Default primary key field type

# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

**URLS.PY**

"""roomonrent URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

    https://docs.djangoproject.com/en/4.0/topics/http/urls/

Examples:

Function views

    1. Add an import:  from my\_app import views

    2. Add a URL to urlpatterns:  path('', views.home, name='home')

Class-based views

    1. Add an import:  from other\_app.views import Home

    2. Add a URL to urlpatterns:  path('', Home.as\_view(), name='home')

Including another URLconf

    1. Import the include() function: from django.urls import include, path

    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import include, path

from ror import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('ror/', include('ror.urls')),

    path('',views.index),

]

**WSGI.PY**

"""

WSGI config for roomonrent project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/4.0/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'roomonrent.settings')

application = get\_wsgi\_application()

**ADMIN**

from django.contrib import admin

from.models import Room

from.models import Feedback

# Register your models here.

admin.site.register(Room)

admin.site.register(Feedback)

**APP**

from django.apps import AppConfig

class RorConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'ror'

**FORMS**

from django.forms import ModelForm

from ror.models import Feedback, Room

class FeedbackForm(ModelForm):

    class Meta:

        model = Feedback

        fields = "\_\_all\_\_"

class RoomForm(ModelForm):

    class Meta:

        model = Room

        fields = "\_\_all\_\_"

**MODELS**

from django.db import models

# Create your models here.

class Room(models.Model):

    room\_image = models.ImageField(upload\_to="static/images")

    room\_address = models.CharField(max\_length=512)

    owner = models.CharField(max\_length=100)

    contact = models.CharField(max\_length=10)

    email = models.EmailField()

    room\_rent = models.IntegerField()

    def \_\_str\_\_(self) -> str:

        return f"{self.owner} {self.room\_address} {self.room\_rent}"

class Feedback(models.Model):

    name = models.CharField(max\_length=100)

    contact = models.CharField(max\_length=10)

    email = models.EmailField()

    massage = models.CharField(max\_length=500)

    def \_\_str\_\_(self) -> str:

        return f"{self.name}"

**URLS**

from django.urls import path

from ror import views

urlpatterns = [

    path('',views.index, name="index"),

    path('index/',views.index, name="index"),

    path('login/',views.log\_in, name="login"),

    path('logout/',views.log\_out, name="logout"),

    path('signup/',views.sign\_up, name="signup"),

    path('feedback/', views.feedback, name="feedback"),

    path('add\_room/', views.add\_room, name="add\_room"),

]

**VIEWS**

from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login\_required

from django.contrib.auth.models import User

from django.contrib.auth import login, logout, authenticate

from django.contrib.auth.hashers import make\_password

from django.contrib import messages

from .models import Room

from ror.forms import FeedbackForm, RoomForm

# Create your views here.

@login\_required(login\_url='/ror/login/')

def index(request):

    rooms = Room.objects.all()

    return render(request,'index.html', {'rooms':rooms})

def log\_in(request):

    if request.method == "POST":

        username = request.POST.get('username')

        password = request.POST.get('password')

        user = authenticate(request,username=username, password=password)

        if user:

            login(request,user)

            return redirect('/ror/index/')

        else:

            messages.add\_message(request, messages.ERROR, "Incorrect username or password!")

    else:

        try:

            if request.user.is\_authenticated:

                return redirect('/ror/index/')

        except Exception as e:

            messages.add\_message(request, messages.ERROR, str(e))

    return render(request,'log\_in.html')

@login\_required(login\_url='/ror/login/')

def log\_out(request):

    logout(request)

    return redirect('/ror/login/')

def sign\_up(request):

    if request.user.is\_authenticated:

            return redirect('/ror/index/')

    if request.method == "POST":

        user = User()

        user.first\_name = request.POST.get('first\_name')

        user.last\_name = request.POST.get('last\_name')

        user.email = request.POST.get('email')

        user.username = request.POST.get('username')

        user.password = make\_password(request.POST.get('password'))

        user.save()

        # if user:

        #     login(request, user)

        #     return redirect('/vault/index/')

        messages.add\_message(request, messages.SUCCESS, "Account created successfully, sign in to continue!")

        return redirect('/ror/login/')

    # else:

    #     try:

    #         if request.user.is\_authenticated:

    #             return redirect('/vault/index')

    #     except Exception as e:

    #         print(e)

    return render(request,'sign\_up.html')

@login\_required(login\_url="/ror/login/")

def feedback(request):

    form = FeedbackForm()

    if request.method == "POST":

        form = FeedbackForm(request.POST)

        if form.is\_valid():

            form.save(commit=True)

            return redirect("/ror/index/")

    return render(request, 'feedback.html', {'form': form})

@login\_required(login\_url="/ror/login/")

def add\_room(request):

    form = RoomForm()

    if request.method == "POST":

        form = RoomForm(request.POST)

        if form.is\_valid():

            form.save(commit=True)

            return redirect("/ror/index/")

    return render(request, 'add\_room.html', {'form': form})

**5. PERFORMANCE ANALYSIS**

**5.1 Testing and Implementing Testing**

The manual testing has been done in the initial phases of the development. The software is working properly as it should be.

An Application testing tool is any program that helps QAs manage and regulate the test process. Deciding which application testing software or framework is to be used varies according to the nature of the application to be tested. Now, let’s explore some of the most popular test automation frameworks used for application testing.

[Selenium](https://www.browserstack.com/selenium) is the most popular tool suite for automating [web application testing](https://www.browserstack.com/web-application-testing-on-cloud/). It enables QAs to verify the [cross browser compatibility](https://www.browserstack.com/guide/cross-browser-compatibility-testing-beyond-chrome) of a web application using[Selenium WebDriver](https://www.browserstack.com/guide/selenium-webdriver-tutorial). Rational Functional Tester (RFT) can be used as an alternative to Selenium.

Following are the fundamental steps involved in testing this application:

1. We created a test plan according to the application requirements
2. Developed manual test case scenarios from the end-users perspective
3. Automated the test scenarios using scripts
4. Performed functional tests and validated and everything worked according to requirements

Application testing is a significant stage in the software development life cycle. Consequently, it becomes necessary for every QA to understand the basics of application testing. This article attempts to foster this understanding so that QAs can do their job in the best possible way.

**5.2 Testing Methods**

           There are two major type of testing they are:

1)      White Box Testing.

2)      Black Box Testing.

**White Box Testing**

White box sometimes called “Glass box testing” is a test case design uses the control structure of the procedural design to drive test case.

Using white box testing methods, the following tests were made on the system

a) All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed

b) All logical decisions were checked for the truth and falsity of the values.

**Black box Testing**

Black box testing focuses on the functional requirements of the software. This is black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors that white box methods like..

1) Interface errors

2) Performance in data structure

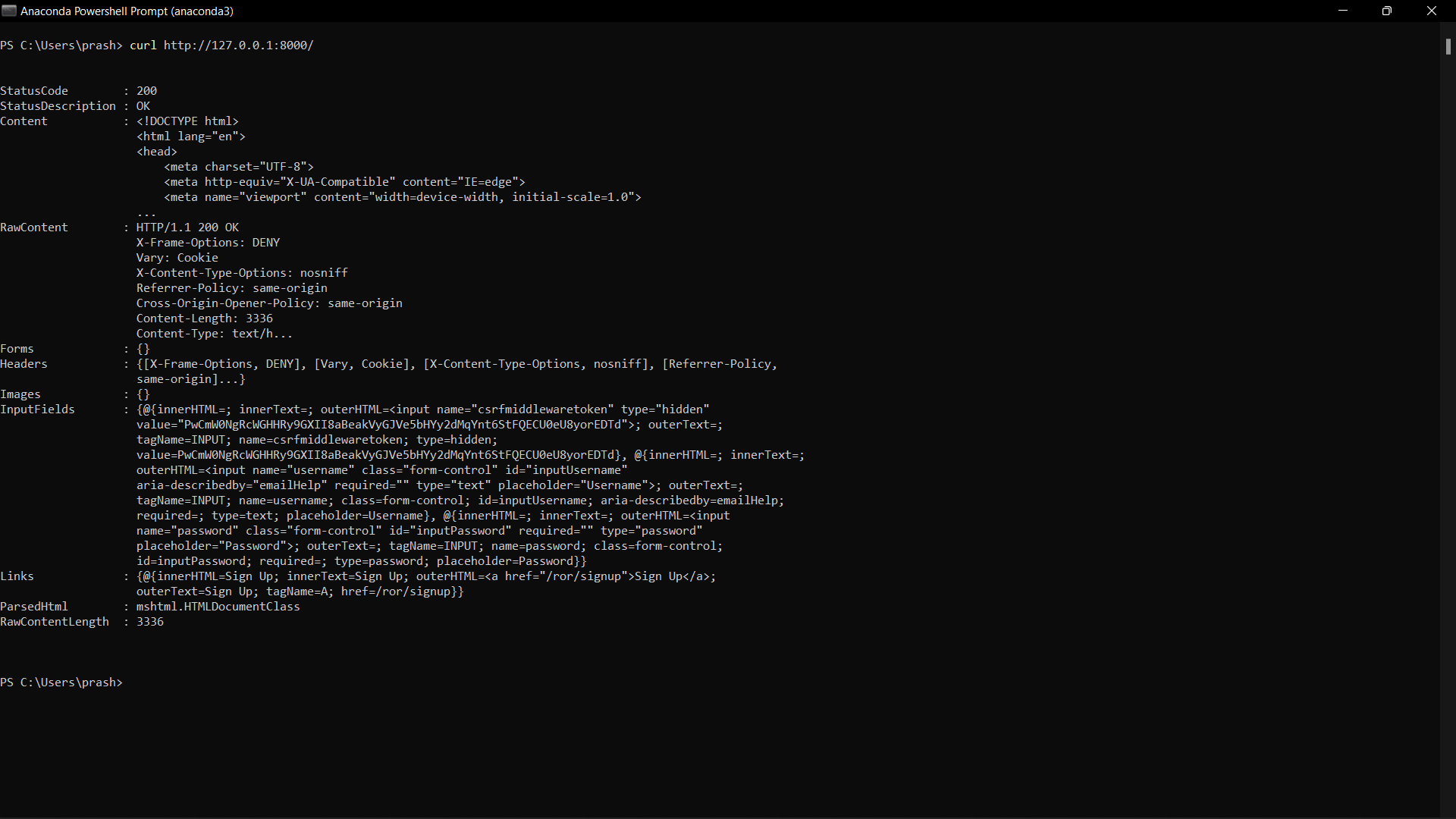
3) Performance errors

4) Initializing and termination errors

**Unit Testing:**

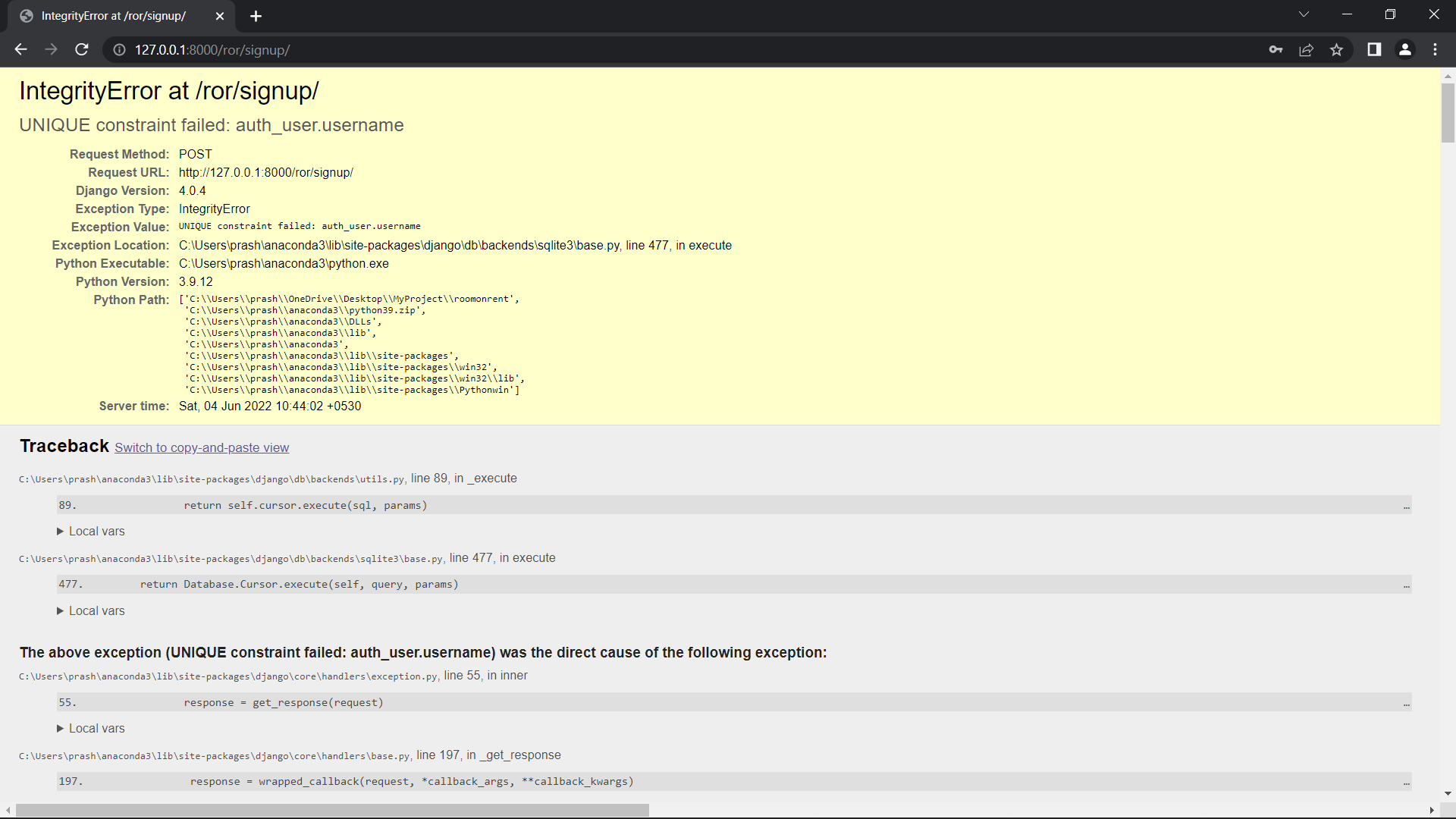
Unit testing is a software verification and validation method in which a programmer tests if individual units of [source code](http://en.wikipedia.org/wiki/Source_code) are fit for use. A unit is the smallest testable part of an application. In [procedural programming](http://en.wikipedia.org/wiki/Procedural_programming) a unit may be an individual function or procedure. Ideally, each [test case](http://en.wikipedia.org/wiki/Test_case) is independent from the others: substitutes like [method stubs](http://en.wikipedia.org/wiki/Method_stub), objects, fakes and [test harnesses](http://en.wikipedia.org/wiki/Test_harness) can be used to assist testing a module in isolation.

This



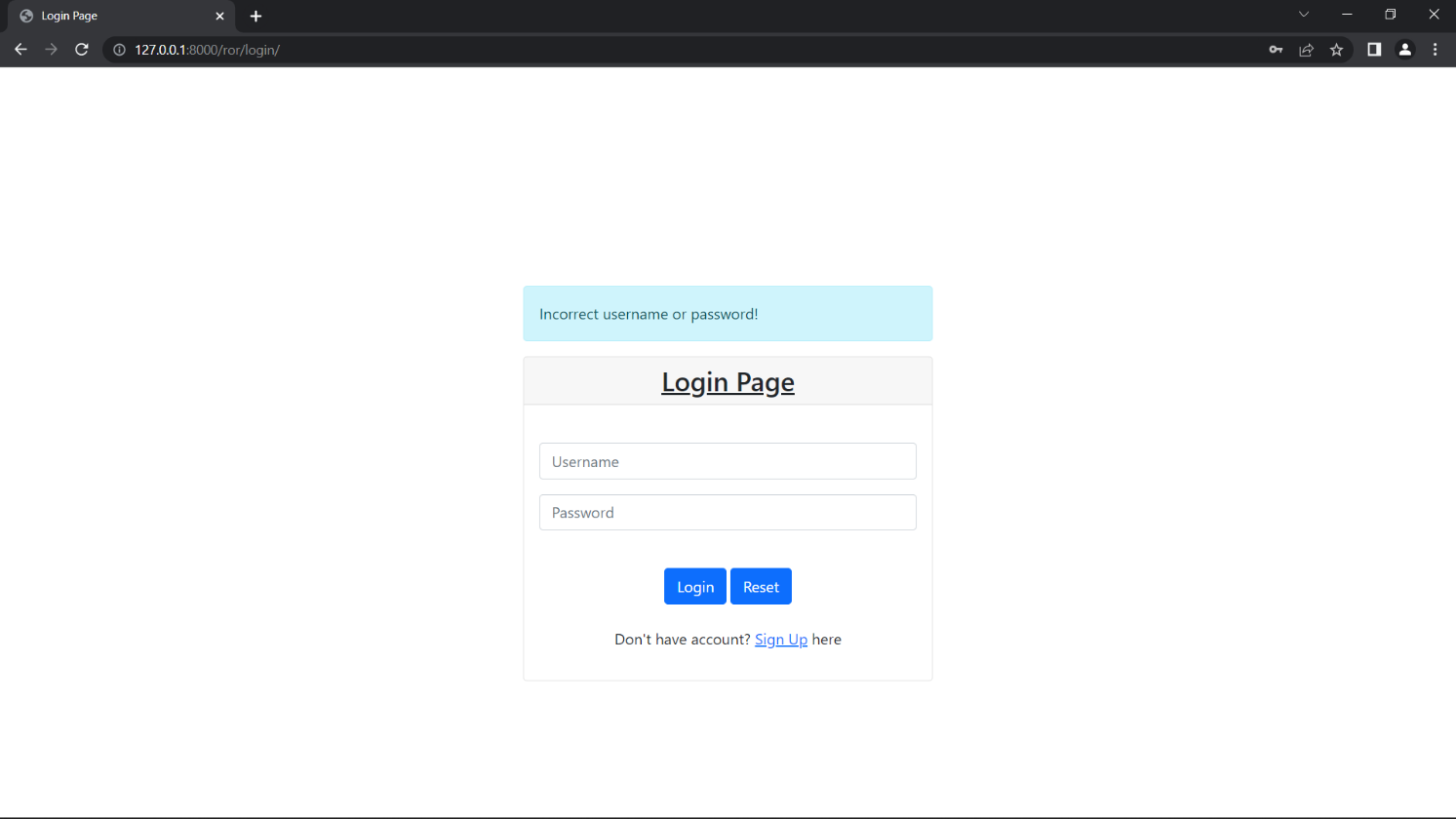
**Integration Testing:**

This testing is sometimes called Integration and Testing. Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.



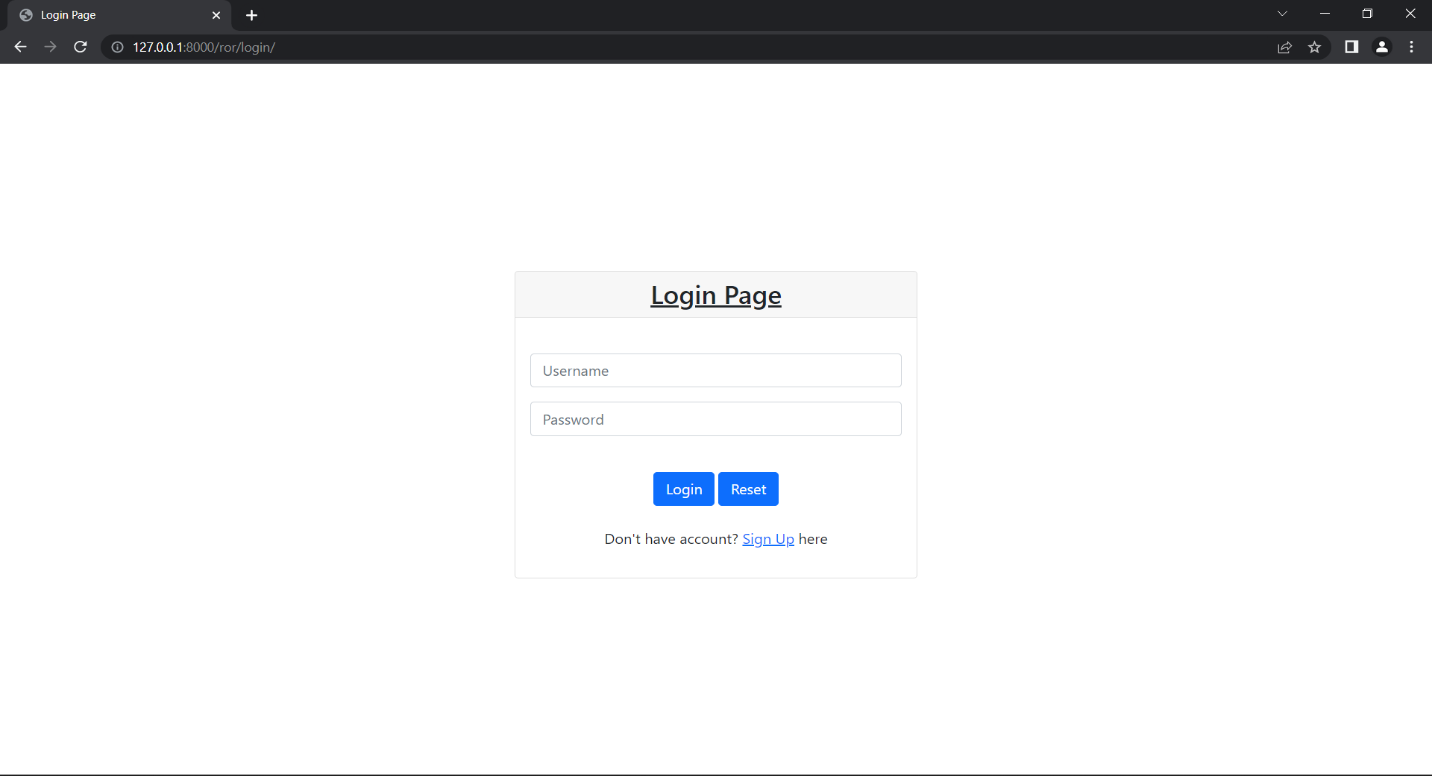
**Validation Testing:**

Validation Testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can reasonably expected by a customer. After validation test has been conducted, one of the following two possible conditions exists. The functions or performance characteristics confirm to specification and are accepted.  
  
• In the administrator and marks modules, all the fields must be filled.  
  
• In the student registration, mobile number should contain exactly 10 numbers.

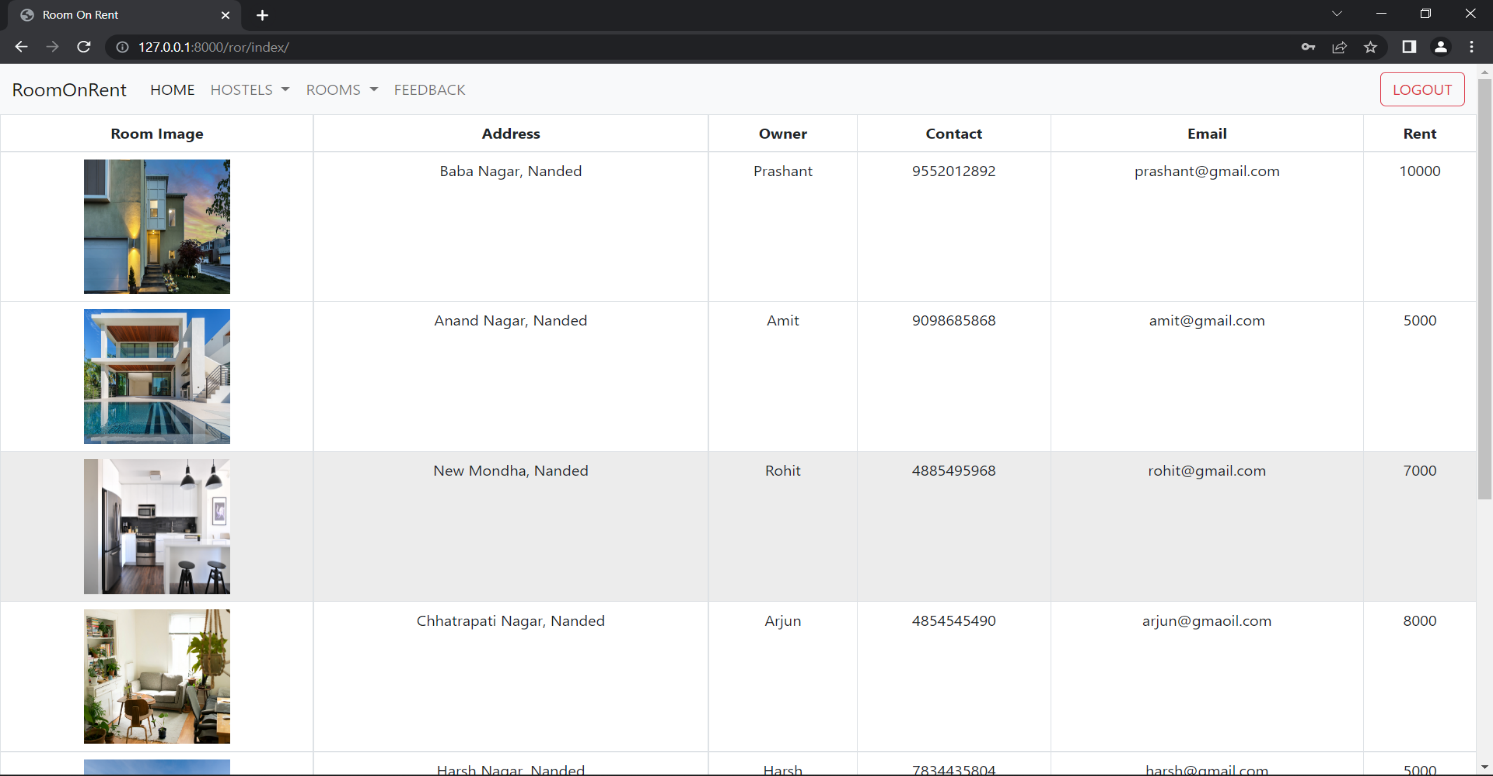


**User Acceptance Testing:**

User acceptance of a system is a key factor of any system. The system under consideration is tested for the acceptance by constantly keeping in touch with the prospective system users at the same time of developing and marketing changes whenever required. This is done in regard to the following points:

**• Input Screen Design**

**• Output Screen Design**



**6. CONCLUSION**

**6.1 Conclusion**

Based on the results, we came up with the following conclusions:

1. The developer were able to create a website that allows the user to search rental rooms and houses, apartment and the like according to their needs.

2. The users can compare room rates.

3. The system can also add new account (owner and clients);

4. The system can show the real-time update of the status and the availability of the rooms;

5. The result shows that both expert and end-user testing got a high score during testing. The results were very positive and the result which can be interpreted as highly usable. Based on the overall result, it is very safe to say that the developed website passed the peoples standard for software development.

**6.2 Future Scope**

With the increasing population across the country, it becomes very difficult to provide home for everyone. According to the Census of India 2011, out of the 90 million residential census units, 11 million units are vacant; that is 12% of the total urban housing stock consists of vacant houses. The need of rental rooms, apartments, houses is very high in India. The large number of students creates a large amount of need of rental rooms. And the future is very much like these rental room apps are going to be very useful as they are now.

**6.3 User Manual**

Login:

It is the first screen that appears when Room On Rent starts. In this form, the user has to enter his/her User ID and Password in order to get access to the site. The authentication process will check the user type and provide the user interface accordingly.

Administrator domain:

The administrator is the super user of this system i.e. he/she will be the first existing user of the system. The administrator has all the access and modification rights.

The following functionalities are provided for the administrator of this system:

**User Information**

In this section, the administrator can add, edit and delete user information. When a new user is added, the system will generate the user id and display it to the administrator.

Administrator adds user information like:

* + - Type

User types are

* + - 1. Administrator
      2. Manager
      3. Staff
      4. User
    - Name

Name of the user.

* + - Password

Password for the user to access the system.

* + - Mobile

Mobile number of the user.

* + - Email

Email address of the user

**Master Settings**

In this section, the administrator has to provide the master details of the rooms which are displayed to rent.

The administrator can add, remove, rename, and change password from here.

**User’s domain**

In this section users can see the images of the rooms. They can see the address of the rooms. They can see the contact information of the room owner.

**6.4 Operations Manual / Menu Explanation**

* **Administrator Domain**

**User Information**

* + 1. Add

To open the Add User Information screen

* + 1. Edit

To open the Edit User Information screen

* + 1. Delete

To open the Delete User Information screen

**Master Settings**

* + 1. Feedbacks

Here the feedback can be seen, and they can be deleted.

ii. Rooms

Here the information of the rooms can be edited, and picture can be edited.

**Change Password**

To open the Change Password Information screen

**Logout**

To exit and terminate current session.

**6.5 Forms and Report Specifications**

Administrator module

1. Add user: The system includes a user form which will collect information about the user. The form will generate a code called as User ID whenever a new user is registered for the system. The ID generated and returned by the system will be given to the user.
2. Add rooms: The system includes a form which will collect the information of the rooms. Here we can add the images of the rooms, information of the owner, and rent rate of the room.
3. Feedback: Here we can see the feedbacks of the users. We can apply crud operations of the feedbacks.

User’s module

Open Welcome Home Windows Page: Here users can see the different types of rooms on the page. They can see the information of the rooms. There are section of boys hostels and girls hostels.

**6.6 Drawbacks and Limitations**

The main drawbacks of the project are as follows:

1. The users cannot book rooms online using this software.
2. This project is only limited to Nanded area only.
3. The map is not available currently.

**6.7 Proposed Enhancements**

Every system has a scope of improvement which when implemented not only increase the efficiency of the system but also provide a new dimension to its capabilities and use.

1. Graphics
2. More features like voice search, videos of room
3. 360 view of rooms
4. Trending page
5. Map system inclusion
6. The services will be made available to larger area.
7. Online room booking will be available.

**REFERENCES**

We would like to mention some sources which proud to be helpful in making this presentation some of them are as follows:

**Book**:-

1) Head first HTML and CSS

- Elisabeth Robson and Eric Freeman

# 2) “Effective PyCharm” by Michael Kennedy and Matt Harrison

**Websites:-**

1) http://www.w3school.com

2) http://www.javatpont.com

3) http://www.geeksforgeeks.org

**ANNEXURES:**

**ACKNOWLEDGEMENT**

I like to share our sincere gratitude to all those who help us in completion of this project. During the work faced many challenges due to my lack of knowledge and experience but these people help us to get over from all the difficulties and in final compilation of our idea to a shaped sculpture.

I wish to express our deep sense of gratitude to our guide, Prof. M. R. Mahamune Dir. S.D.Khamitkar (H.O.D of computational department SRTM University, Nanded) for his valuable guidance, keen interest, constructive suggestions and sustained encouragement throughout the project work.

I am also thankful to my whole class and to my parents who have inspired me to face all challenges and win all hurdles in life.

**Members Name**

Yeotikar Prashant S.

------------------------

Patil Ashutosh A.

------------------------